A PRIMERA REVISTA DE PROGRAMACIÓN EN CASTELLANO

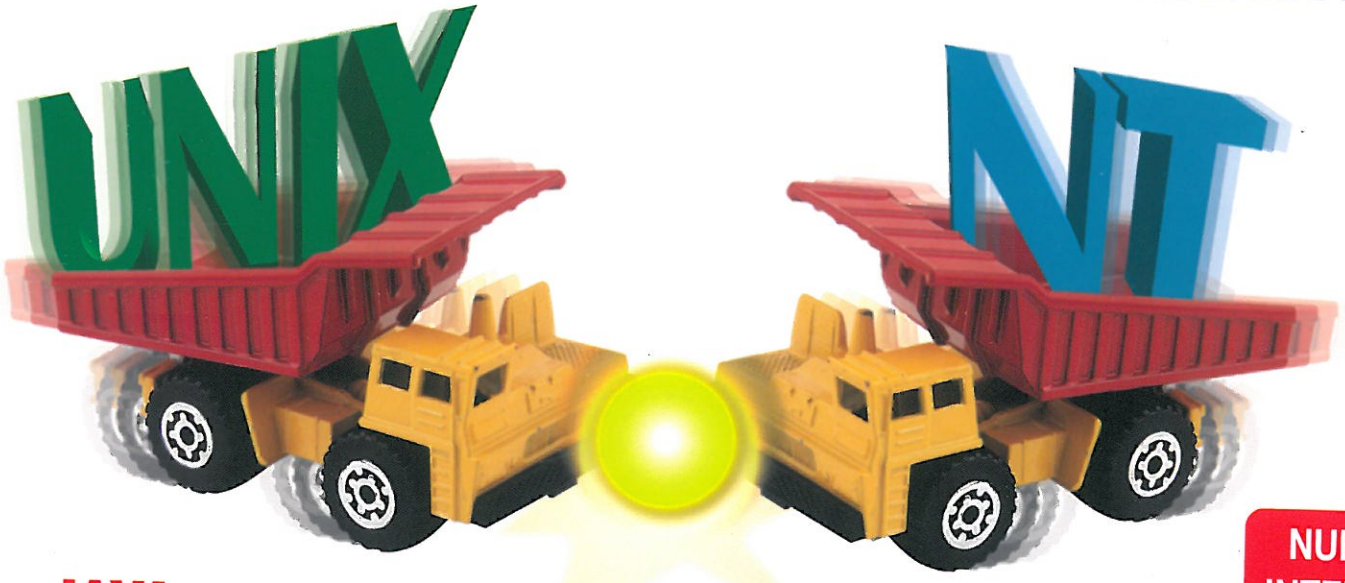
SOLO PROGRAMADORES

) V, NÚMERO 50

975 Ptas.

UNIX VS WINDOWS NT

A GUERRA DE LOS SISTEMAS OPERATIVOS



JAVA
Concurrencia y sincronización de threads (YIII)

LINUX SVGALIB Y (IV)
El último escollo, manejo de periféricos

SEGURIDAD
Protección de servidores y redes (I)

MULTIMEDIA
Reconocimiento de voz (III)

NUEVO INTERFAZ

- Active Multimedia button 2.0
- Netstudio 1.0
- Setup Specialist 98

CONTENIDO DEL CD-ROM

VISUAL BASIC
Programación del API de Windows (IV)

C++ BUILDER 3
Creación de componentes (Y III)

Active X, Animaciones, CGI, Correo electrónico, Diseño gráfico, FTP, FrontPage, Html, Html Dinámico, Java, Java Script, Navegadores, Visual J ++.

PREPÁRATE PARA SER EL MEJOR WEBMASTER

DESCRIPCIÓN DE LA COLECCIÓN:

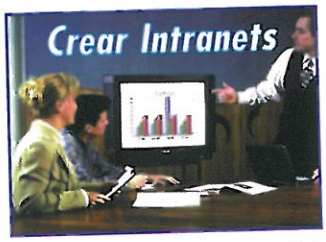
- 5 CD ROM. Con los programas necesarios para el seguimiento del curso y el código de los ejercicios expuestos en el mismo.
- 500 fichas a todo color.
- 4 carpetas rígidas con anillas.
- 1 carpeta para los disquetes y CD-Roms.
- 48 disquetes.

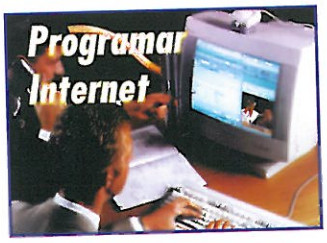
Contenidos prácticos con ejemplos, ejercicios, casos, modelos, guías, etc...
Se entregará un diploma acreditativo a las personas que finalicen el curso.

GRATIS Versión Oficial con licencia

Valorado en 22.270 Ptas

Aprenderás a:





curso WEBMASTER de programación Internet

Por sólo 29.990 PTAS

Solicítanos información en el tfno. 91-6614211, en el correo electrónico suscrip@towercom.es o envíanos este cupón y te contaremos cómo suscribirte y cómo funcionará la primera Bolsa de trabajo de expertos en Internet relacionada con este Cur

---

Nombre y Apellidos ..... F. nacimiento .....
Domicilio ..... C.P. .....
Ciudad ..... Provincia ..... Telf.: .....
e-mail ..... Profesión .....
Apartado de correos ..... Código Postal .....

☐ Deseo recibir amplia información sobre el Curso Webmaster

PROGRAMADORES

Número 50
SÓLO PROGRAMADORES
es una publicación de
TOWER COMMUNICATIONS

Director Editor
Antonio M. Ferrer Abelló
aferrer@towercom.es
Coordinador Técnico
Eduardo De Riquer Frutos
eriquer@towercom.es
Redactora Jefe
Cristina Peña
crisp@towercom.es
Colaboradores
Constantino Sánchez Ballesteros, Jorge Figueroa, Juan J. Taboada, Juan J. Mora, Jordi Agost, Javier Toledo, Jorge Delgado, Javier Sanz, Francisco M. Rosado.
Maquetación
María Arce, Javier Pérez, M.A Gálvez
Tratamiento de Imagen
María Arce Giménez

Publicidad
Erika de la Riva (Madrid)
Tel.: (91) 661 42 11
Pepín Gallardo (Barcelona)
Tel.: (93) 213 42 29 - 970 47 44 35

Suscripciones
Begoña Álvarez
Tel. (91) 661 42 11 Fax: (91) 661 43 86
suscrip@towercom.es

Laboratorio
Óscar Rodríguez (jefe)
oscarrf@towercom.es
Javier Amado
jamado@towercom.es
Servicio Técnico
Óscar Casado
ocasado@towercom.es

Preimpresión
Indes Color
Impresión
Gráficas Muriel
Distribución
SGEL
Distribución en Argentina / Chile / Colombia / México / Venezuela
Capital: Huesca y Sanabria
Interior: D.G.P.

TOWER COMMUNICATIONS
Director General
Antonio M. Ferrer Abelló
Director Financiero
Francisco García Díaz de Liaño
Director de Producción
Carlos Peropadre
Directora Comercial
Carmina Ferrer
carmina@towercom.es

Redacción, Publicidad y Administración
C/ Aragoneses, 7
28108 Pol. Ind. Alcobendas (MADRID)
Telf.: (91) 661 42 11 / Fax: (91) 661 43 86

La revista Sólo Programadores no tiene por qué estar de acuerdo con las opiniones escritas por sus colaboradores en los artículos firmados. El editor prohibe expresamente la reproducción total o parcial de los contenidos de la revista sin su autorización escrita.

Depósito legal: M-26827-1994
ISSN: 1134-4792
PRINTED IN SPAIN
COPYRIGHT 30-1-99

Crackeando el talento

Es posible que no me crean, sobre todo con los acontecimientos vertiginosos que suceden en nuestro sector, pero pocos aspectos técnicos tienen una relación causa-efecto más acusada que los famosos *cracks*. La capacidad de reacción de sus creadores siempre ha sido más que notable, pero el impulso definitivo, cómo no, se ha producido de la mano de Internet. No comprendo si se trata de una competición entre los grupos que se dedican a esta "distinguida" tarea o si los realizan desde una posición de privilegio (que alguien se atreva a decir que nunca ha pensado que muchos virus proceden de las propias empresas desinfectadoras). En cualquier caso, la realidad confirma la existencia de servidores en Internet que contienen largas y completas listas con los *crackers* disponibles para cualquier programa comercial imaginable, sin olvidar las numerosas versiones beta que aparecen durante el desarrollo del producto.

Realmente uno se pregunta cómo es posible "desperdiciar" tanto talento, sin saber a ciencia cierta qué ocultos intereses mueven los engranajes del asunto: ego personal, competición con otros compañeros del gremio o rastreros intereses económicos. Menos en el último supuesto, estos personajes me recuerdan a una especie de Robin Hood, dedicado a robar los bits a las empresas ricas para ponerlos al alcance de los pobres usuarios.

Entiendo que lo anterior pueda malinterpretarse, pero no creo que las pérdidas de las empresas de software (o mejor dicho, las ganancias adicionales no obtenidas, porque nunca cierran con déficit), se deban a los usuarios personales. Si éstos hubieran tenido que pagar por cada programa utilizado con los precios de costumbre, creo que sencillamente aún se utilizaría la máquina de escribir.

Es cuando se habla de empresas cuando la cuestión cambia de forma radical, puesto que además de tratarse de una actitud ilegal, el grado de inmoralidad en estos casos resulta vergonzoso. Volviendo a nuestro Robin Hood particular, tendríamos a un príncipe que roba a los ricos para hacerse un lujoso castillo, y en estos casos la actuación de la justicia debería ser implacable.

En otro extremo nos encontramos con otro grupo de talentos, que a mi juicio, sí que resultan inclasificables, por no utilizar otro término: los creadores de virus. Me cuesta mucho creer que la libertad de expresión pueda esconder personalidades tan soberbias, cuya aportación final consiste en dañar, entorpecer, inutilizar, arruinar y mil sinónimos más, el trabajo de otros usuarios anónimos y desconocidos.

Por este motivo y dentro de los grupos comentados (al margen de mercenarios), me quedo con los que "desprecian" su talento acercando la informática a los usuarios corrientes. Sí, porque cuando se trata de usuarios de empresas, resulta difícil establecer algún tipo de distinción entre héroes y villanos.

SOLO PROGRAMADORES

50

8 Noticias
ULTIMAS NOVEDADES

Todo lo relacionado con las nuevas tecnologías del mercado informático llegan hasta estas primeras páginas para que nuestros lectores estén al día de lo que ocurre en cuanto a productos y alianzas entre empresas informáticas se refiere. Todas las novedades que te interesan para estar al día en materia de hardware.

8 Java
PROGRAMACIÓN EN JAVA USANDO THREADS (Y III)

Damos por finalizada esta serie de artículos con esta tercera entrega donde nos adentraremos en la creación de grupos de threads, estudiando los métodos más útiles para poder trabajar con ellos, así como el desarrollo de applets de forma sencilla gracias a los hilos.

17 Linux
SVGALIB (Y IV): EL ÚLTIMO ESCOLLO, MANEJO DE PERIFÉRICOS

Una vez vistos los tres anteriores capítulos, SVGALI no debe tener ni un solo secreto para nuestros lecto res. En esta última entrega nos vamos a dedicar íntegramente a bucear en el manejo de periférico.

24 C/C++
PROGRAMACIÓN BÁSICA EN WINDOWS (IV)

Trataremos de poder incorporar diversos recursos Windows a nuestras aplicaciones personales, tales como iconos, cursores, nuestros propios tipos de letra gráficos, sonidos, etc.

37 Sistemas Operativos
UNIX VS WINDOWS NT

Existe una pregunta que muchos de nuestros lectores se habrán hecho más de una vez: ¿cuál es el mejor sistema operativo?. No existe una respuesta única, ya que la contestación depende en gran medida de las necesidades personales de cada usuario, así como de los criterios propios. Unix y Windows NT tienen sus propias ventajas e inconvenientes. Atrévete a descubrirlos.

32 Visual Basic
PROGRAMACIÓN DEL API DE WINDOWS (III)

Ahora es el momento ideal para que el lector se introduzca en los ejemplos prácticos sobre el entorno de la API, una vez vista la teoría en las anteriores entregas. Los tipos de menús que existenten, las funciones con el ratón o las capturas en pantalla, serán algunos de nuestros objetivos a lo largo de estas páginas.

52 Programación Multimedia
RECONOCIMIENTO DE VOZ (III)

Esta apasionante serie continúa descubriéndonos nuevos aspectos dentro del sorprendente mundo de la programación multimedia. Tras dos capítulos anteriores, seremos capaces, a partir de ahora, de abordar el primer proyecto de reconocimiento de comandos por medio de la API Speech, metiéndonos de lleno en la práctica.

67 C++ Builder 3
CREACIÓN DE COMPONENTES (Y III)

Tercera y última entrega de esta serie donde no dejaremos escapar la ocasión de aprender a crear un icono propio, un componente ShareWare, paso a paso sin que surja ningún tipo de problemas, de un modo totalmente sencillo.

74 Libros
MANUALES PARA TODOS

Un mes más os acercamos ejemplares que os puedan ayudar en vuestras aplicaciones. Pensando en todos vosotros, presentamos la Biblia de Delphi 3, Visual Basic 5.0, Manual fundamental de Perl 5 y Windows NT Server 4. Todos ellos os proporcionarán la actualidad de estas materias.

58 Seguridad
PROTECCIÓN DE SERVIDORES Y REDES EN INTERNET (I)

Hoy en día todo el mundo quiere conectarse a la red y con el crecimiento de esta demanda ha aumentado el peligro para la información que en ella circula. A partir de este planteamiento es cuando se buscan las medidas de seguridad y protección, más conocidas como firewalls o cortafuegos, para mantener la privacidad.

76
CORREO DEL LECTOR

En esta sección se pretende dar respuesta a todas las dudas que puedan tener nuestros lectores, pudiendo dirigirse a los propios autores o bien a la siguiente dirección de correo de nuestra publicación: solop@towercom.es.

78
CONTENIDO DEL CD-ROM

Un mes más un nuevo CD-ROM complementa "Sólo Programadores" incluyendo todo aquello que os pueda servir de ayuda y complemento a los artículos que aparecen en nuestras páginas. En esta ocasión hallaréis programas de gran utilidad como Active Multimedia Button 2.0 o NetStudio 1.0 entre otros.

Sun anuncia una nueva tecnología para aplicaciones empresariales colaborativas

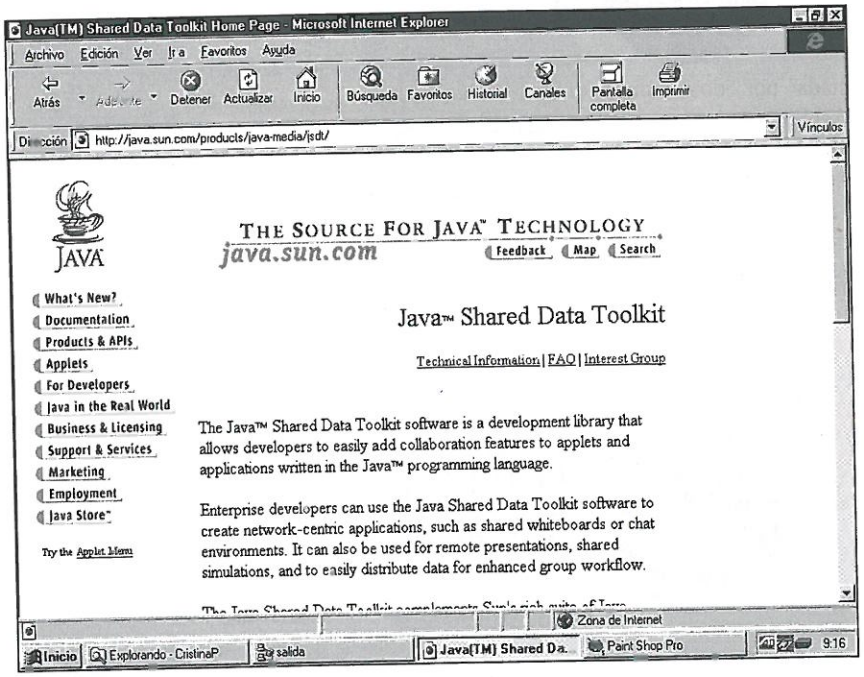
JAVA SHARED DATA TOOLKIT YA ESTÁ DISPONIBLE

La compañía Sun Microsystems ha anunciado la disponibilidad de la tecnología Java Shared Toolkit, un conjunto de herramientas orientado a la creación de applets y aplicaciones colaborativas distribuidas con un alto grado de interacción basadas en lenguaje de programación Java.

El software Java Shared Data Toolkit lo pueden incorporar los desarrolladores a aplicaciones y herramientas de desarrollo para la creación de foros de conversación, aplicaciones compartidas, presentaciones remotas u otras aplicaciones que precisen la incorporación de funcionalidades colaborativas.

El software Java Shared Data Toolkit satisface los estándares de certificación 100% pure Java de Sun, por lo que se puede utilizar en cualquier plataforma que soporte el software Java Development Kit (JDK)1.1.

En este software se incluyen ejemplos como las aplicaciones que demuestran las funcionalidades de colaboraciones diferentes como: un entorno de conversación donde los participantes pueden teclear sus conversaciones; una aplicación que permite a los usuarios compartir diagramas o dibujos; un visualizador que permite consultar las actualizaciones continuas de las cotizaciones facilitadas por los teletipos, etc.



NUEVA CAMPAÑA DE PERVASIVE

La compañía Pervasive acaba de anunciar una campaña completa dirigida a la Comunidad Mundial de Usuarios NetWare con el objetivo de satisfacer la creciente demanda existente de la nueva tecnología de acceso paralelo ofrecida con su último producto que actualmente ofrece al mercado Pervasive SQL.

La tecnología de acceso paralelo ha sido anunciada junto con Pervasive SQL, la única solución de base de datos que ofrece tanto el acceso de datos transaccional como relacional desde un motor compartido.

Puede visitar la dirección web www.pervasive.com

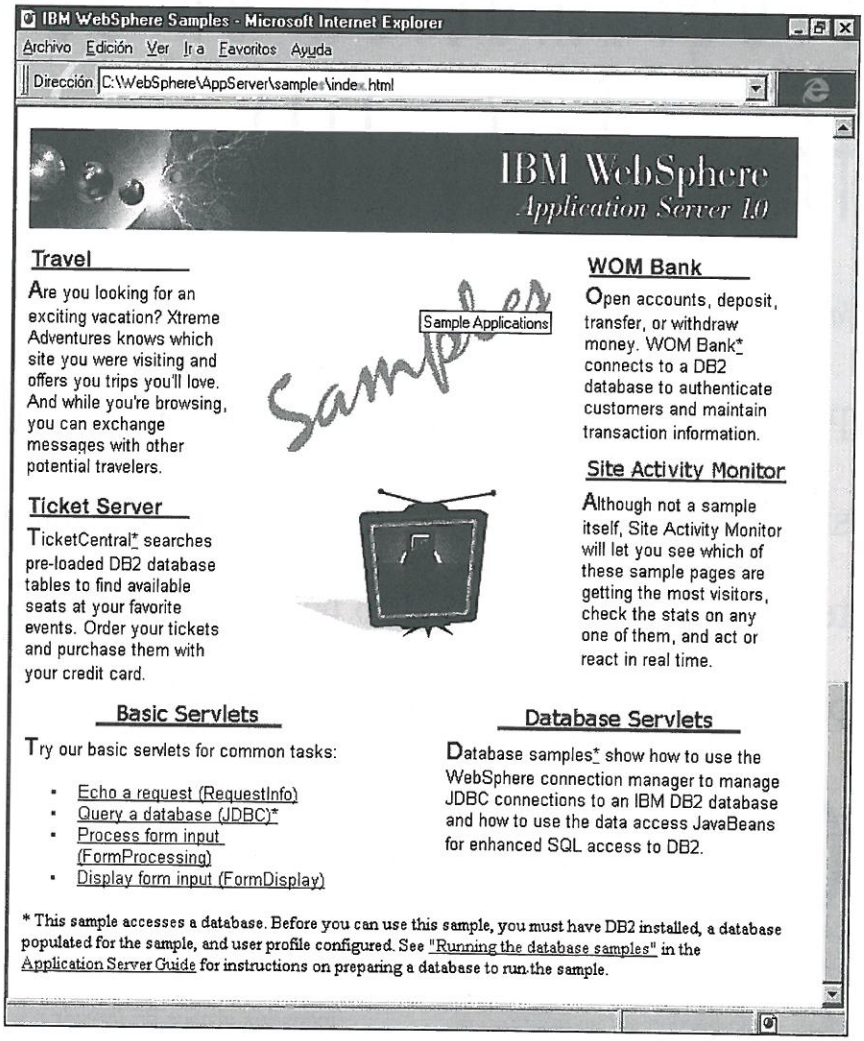
IBM ANUNCIA SOFTWARE PARA CREAR Y GESTIONAR APLICACIONES WEB

IBM ha anunciado una nueva línea de productos de software denominada IBM WebSphere, que proporciona todo lo necesario para crear, implementar y gestionar aplicaciones Web basadas en Java. IBM WebSphere permite a las empresas de todos los tamaños tener una mayor presencia en la web y beneficiarse de las ventajas del modelo e-business.

La familia de productos está formada por dos componentes: IBM WebSphere Application Server e IBM WebSphere Performance Pack. A finales de este año la compañía ampliará esta línea de productos con un conjunto integrado de herramientas para construir aplicaciones Java dinámicas que puedan ser instaladas y gestionadas en Internet, intranet o extranets.

IBM WebSphere Application Server ofrece a los desarrolladores una base flexible y abierta para implementar aplicaciones Web utilizando Java.

IBM WebSphere Performance Pack es un software de gestión Web que proporciona un entorno fiable y escalable, preparado para soportar el rápido crecimiento de páginas web.



ORACLE PRESENTA ORACLE ENTERPRISE DEVELOPER SUITE 2.1

La compañía Oracle anuncia el lanzamiento de Oracle Enterprise Developer Suite Release 2.1, que es una actualización del paquete de aplicaciones de desarrollo y servidores de la compañía.

El producto está compuesto por Oracle Designer 2.1 y Oracle Developer 2.1. Esta suite se convierte en el primer kit de herramientas especializado en el diseño de bases de datos con la Network Computing Architecture (NCA) de Oracle.

La versión 2.1 de Oracle Designer (herramienta de modelado) presenta un interfaz renovado, además de reducir considerablemente los costes de administración ya que permite desarrollar aplicaciones completamente adaptables, controlar las aplicaciones de los servidores a los que se accederá desde un PC y un explorador web.

Puede obtener más información en la página web www.oracle.es

Programación en Java usando threads (y III)

Javier Sanz Alamillo (jsanz@a01-unix.uc3m.es)

Para efectuar determinadas acciones sobre un conjunto de threads de forma simultánea se pueden crear grupos de hilos, de manera que no hay necesidad de tener que llamar a un método por cada uno de ellos. Gracias a los threads, se pueden desarrollar applets de forma sencilla con unos resultados espectaculares.

Cuando una aplicación utiliza multihilos siempre surge la necesidad de realizar ciertas operaciones que afecten a todos los hilos en ejecución. Es más, en determinadas condiciones también es necesario agrupar estos *threads* según ciertas características, como pueden ser su prioridad de ejecución, de forma que determinados cambios afecten a los hilos de según que grupos.

GRUPOS DE THREADS

En Java, todos los *threads* pertenecen a un determinado grupo, el cual puede puede ser el grupo por defecto determinado por el sistema o un grupo que el programador haya definido de forma explícita. Durante el proceso de creación el *thread* se añade a un grupo y no puede cambiar durante su vida a otro diferente. Si se crean más de un *thread* sin especificar su grupo estos serán añadidos al grupo del sistema.

Todos los grupos de *threads* a su vez pertenecen a otro grupo de *threads*. Un nuevo grupo puede pertenecer a otro según se defina en el constructor del grupo de *threads* o si no se especifica nada, se le incluirá como un grupo hijo dentro del grupo del sistema.

La razón de la existencia de los grupos de *threads* para cada grupo, es decir, el motivo por el que un grupo siempre debe estar incluido en otro es de difícil explicación. La mayoría de las publicaciones explican que esta característica viene dada por razones de seguridad, ya que los *threads* de un grupo pueden modificar a otros *threads* en otro grupo, incluyendo a los del grupo padre. Pero los *threads* que pertenecen a un grupo no pueden modificar a otros que se encuentren en un grupo con el que no mantienen relación, ya sea por ser otro grupo padre, o por no ser un grupo hijo directo.

MÉTODOS PARA LOS GRUPOS DE THREADS

Los métodos utilizables para los grupos de threads se pueden clasificar en función de:

- Si actúa sobre los threads que pertenecen a un grupo.
- Si la acción recae sobre los thre-

ads que pertenecen al grupo actual y a los grupos relacionados.

En el primer caso tenemos por ejemplo los métodos setDaemon() y setMaxPriority() que no modifican las características del grupo, sino la de los sus threads. En cambio los siguiente métodos:

resume()
stop()
suspend()

Son los que actúan sobre todos los threads del grupo y de sus grupos descendientes.

Además tenemos que destacar que los siguientes métodos permiten obtener los atributos de un grupo de threads:

getMaxPriority()
getDaemon()
getName()
getParent()
list()

De estos métodos, getName() devuelve el nombre de un determinado thread para poder realizar determinadas acciones sobre uno particular. El método getParent() devuelve el grupo padre al que pertenece el grupo del thread actual. El método list() es un método especial pues permite visualizar las características de un grupo de threads, por lo que se usa principalmente en la depuración de código.

EJEMPLOS PRÁCTICOS

Una vez que conocemos los fundamentos de los grupos de threads vamos a desarrollar unos ejemplos para poner en práctica la teoría. En el ejemplo del Listado 1 vamos a ver como los threads de un determinado grupo

Listado 1. Utilización de grupos de threads hijos.

```
public class acceso {
public static void main ( String args[] ) {
// Se definen los grupos de threads
ThreadGroup x = new ThreadGroup ("x");
ThreadGroup y = new ThreadGroup (x,"y");
ThreadGroup z = new ThreadGroup (y,"z");

// Generamos un par de hilos de prueba
Thread h1 = new usaThread1 (x, "hilo1"), h2 = new usaThread2 (z, "hilo2");
 }
}
class usaThread1 extends Thread {
usaThread1 ( ThreadGroup grupo, String nombre )        {
super ( grupo, nombre );
 }
void f()       {
System.out.println ( "el hilo : " + getName() + " ejecuta f() ");
  }
}
class usaThread2 extends usaThread1 {
usaThread2 ( ThreadGroup grupo, String nombre )        {
  super ( grupo, nombre );
  start();
}
  public void run() {
ThreadGroup grupo = getThreadGroup().getParent().getParent();
       grupo.list();
       Thread [] todos = new Thread [ grupo.activeCount() ];
       grupo.enumerate ( todos );
       for ( int i = 0 ; i < todos.length; i ++ )      {
            todos[i].setPriority ( Thread.MIN_PRIORITY);
            ( (usaThread1) todos[i]).f();
       }
      grupo.list();
  }
}
```

"hijo" pueden acceder a los métodos de los threads del grupo padre.

Ejecutamos el código y obtenemos a continuación:

```
java.lang.ThreadGroup[name=x,maxpri=10]
  Thread[hilo1,5,x]
  java.lang.ThreadGroup[name=y,maxpri=10]
    java.lang.ThreadGroup[name=z,maxpri=10]
       Thread[hilo2,5,z]
el hilo : hilo1 ejecuta f()
el hilo : hilo2 ejecuta f()
java.lang.ThreadGroup[name=x,maxpri=10]
  Thread[hilo1,1,x]
  java.lang.ThreadGroup[name=y,maxpri=10]
    java.lang.ThreadGroup[name=z,max-
    pri=10]
      Thread[hilo2,1,z]
```

En el código se observa que en la clase acceso se definen los grupos de threads. En la salida del programa se aprecia que el grupo <x> es un grupo que pertenece al sistema, por lo que hereda las características del mismo. Como vemos en la primera línea dicho grupo contendrá hilos con una priori-

dad máxima de valor 10. Los siguiente grupos, tales como el <y> que será un subgrupo de <x> y a su vez, <z> que lo será de <y>.

Los dos threads que se crean pertenecerán respectivamente a los grupos <y> y <z>. En la clase de usaThread1 definimos los métodos para los threads del grupo <y> y a su vez, en la clase usaThread2 para los de <z>, que heredan los métodos del grupo de threads <y>.

En la ejecución del método run() de usaThread2 se crean los hilos del grupo y se visualizan sus características. A continuación se crea una tabla con todos los threads que están en ejecución con ayuda del método activeCount() que devuelve el número de hilos activos.

Mediante el método enumerate() se copian en la tabla especificada el conjunto de los threads de un determinado grupo y su primer nivel de subgrupos. Existe un método sobrecargado de éste que permite incluir a todos los grupos "hijos" del grupo en cuestión de forma recursiva.

Ya que tenemos los threads en una tabla, les cambiamos la prioridad y hacemos que llamen a la función f() de la clase usa Thread1, independientemente de si los threads pertenecen al grupo padre o hijo.

Con este ejemplo que hemos expuesto ha quedado reflejada la característica que indica que los grupos de threads hijos pueden acceder a los métodos del grupo padre. Pero la característica más importante del grupo de threads es la facilidad del control de los mismos en forma de conjunto, cuando se pretende que una determinada acción afecte a todos los threads.

Con el ejemplo del Listado 2 se demuestra lo sencillo que resulta manipular las propiedades de los grupos de threads, así como la facilidad con la que

Listado 2. Manipulación de las propiedades de los grupos de threads.

```
public class control
{
   public static void main ( String args[] )
   {

      // (1) Se buscan todos los thread del grupo      // principal
      ThreadGroup sistema = Thread.currentThread().getThreadGroup();
      sistema.list();

      // (2)Cambiamos la prioridad del grupo sistema
      sistema.setMaxPriority( Thread.MAX_PRIORITY - 1 );
      sistema.list();

      // (3)Modificamos la prioridad del thread main
      Thread main = Thread.currentThread();
      main.setPriority( main.getPriority() + 1);
      sistema.list();

      // (4)Nuevo grupo
      ThreadGroup g1 = new ThreadGroup ("Grupo 1");
      g1.setMaxPriority( Thread.MAX_PRIORITY );
      g1.list();

      // (5) Comprobacion de prioridad
      Thread hilo1_g1 = new Thread ( g1, "hilo1_g1");
      hilo1_g1.setPriority( hilo1_g1.getPriority() + 1);
      g1.list();

      // (6) Cambio en prioridad
      Thread hilo2_g1 = new Thread ( g1, "hilo2_g1");
      hilo2_g1.setPriority( Thread.MAX_PRIORITY );
      g1.list();

      // (7) Vamos la prioridad que va a heredar
      g1.setMaxPriority( Thread.NORM_PRIORITY );
      g1.list();

      // (8) Nuevo grupo h2 que es hijo de g1
      ThreadGroup h1 = new ThreadGroup (g1, "Hijo de Grupo 1");
      h1.setMaxPriority( Thread.MAX_PRIORITY );
      h1.list();

      // (9) Creacion de hilos
      for ( int x = 0 ; x < 4 ; x ++ )
            new Thread ( h1, "hilo"+Integer.toString(x)+"_h1");
      h1.list();

      // (10) Visualizar todos los threads y pararlos .
      Thread [] todos = new Thread [ sistema.activeCount()];
```

Listado 2. (Continuación).

```
        sistema.enumerate(todos);
        for ( int x = 0 ; x < todos.length ; x ++ )
        {
                if ( ! todos[x].isAlive())
                        todos[x].start();
        }
        // (11) Control de los threads
        //System.out.println (" Todos los thread estan dormidos ..");
        //sistema.suspend ();
        // No llegara por aqui si todos duermen ...
        System.out.println (" Todos los thread van a morir ..");
        sistema.stop();
    }
}
```

todos los threads de un grupo realizan una misma función mediante una simple llamada a un método.

De la ejecución del código obtenemos el siguiente resultado:

```
java.lang.ThreadGroup[name=main,maxpri=10]
   Thread[main,5,main]
java.lang.ThreadGroup[name=main,maxpri=9]
   Thread[main,5,main]
java.lang.ThreadGroup[name=main,maxpri=9]
   Thread[main,6,main]
java.lang.ThreadGroup[name=Grupo 1,max-
      pri=9]
java.lang.ThreadGroup[name=Grupo 1,max-
      pri=9]
   Thread[hilo1_g1,7,Grupo 1]
java.lang.ThreadGroup[name=Grupo 1,max-
      pri=9]
   Thread[hilo1_g1,7,Grupo 1]
   Thread[hilo2_g1,9,Grupo 1]
java.lang.ThreadGroup[name=Grupo 1,max-
      pri=5]
   Thread[hilo1_g1,7,Grupo 1]
   Thread[hilo2_g1,9,Grupo 1]
java.lang.ThreadGroup[name=Hijo de Grupo
      1,maxpri=5]
java.lang.ThreadGroup[name=Hijo de Grupo
      1,maxpri=5]
   Thread[hilo0_h1,5,Hijo de Grupo 1]
   Thread[hilo1_h1,5,Hijo de Grupo 1]
   Thread[hilo2_h1,5,Hijo de Grupo 1]
   Thread[hilo3_h1,5,Hijo de Grupo 1]
Todos los thread van a morir ..
```

Después de observar esta segunda ejecución analizaremos los resultados obtenidos.

De la ejecución de la parte (1) lo que obtenemos son los threads del grupo del sistema, por lo que también se obtienen en consecuencia sus características principales.

En la secuencia (2) se cambia la prioridad del grupo "sistema", afecta a la prioridad máxima que pueden obtener los threads de ese grupo.

En (3) se cambia la prioridad del thread main del grupo sistema.

En la parte (4) se genera un grupo hijo de sistema, y se le adjudica la máxima prioridad, que es la determinada por el grupo sistema.

En (5) y (6) se generan dos hilos para este último grupo y se observa la forma en que se puede modificar el atributo de prioridad.

En (7) se comprueba la herencia del atributo prioridad según viene determinado por el grupo padre y en (8) se crea un nuevo grupo hijo del anterior. La creación de varios threads para este grupo y la visualización de sus propiedades ocurre en (9) y (10).

En el apartado (11) se puede ejecutar la parte de suspend() que cambiará a estado de dormido a todos los threads del grupo y comentado esas líneas, se ejecutaría el stop() que finalizaría todos los threads creados.

Como se ha podido comprobar con estos ejemplos que hemos expuesto, el manejo de grupos de threads es simple y eficaz, de forma que el control de los grupos de threads ofrece una mayor potencia que el manejo individual de cada uno de ellos.

LOS THREADS EN LOS NAVEGADORES

Actualmente es muy común encontrar en las páginas web en Internet programas que se ejecutan en el navegador. Estos programas son applets Java, y su desarrollo esta íntimamente ligado al desarrollo de threads. La estructura básica de un applet recuerde a la implementación de un thread, de forma que se obtiene:

```
public class miApplet extends Applet imple-
      ments Runnable {
}
```

Por tanto, estamos definiendo ya nuestro applet como un thread, de tal forma que sólo queda implementar el método run() que desarrolle la tarea que deseemos obtener. Además, los applets ofrecen características propias, disponiendo de unos procedimientos que se ejecutan automáticamente cuando el applet se carga y se destruye. Por ejemplo, el método init() es el primer método que se ejecuta cuando se carga un applet. Éste a su vez llama al método start() y es ahí donde comienza la ejecución propiamente dicha, desde donde se llama a los métodos update() y paint() para la visualización en panta-

lla. La parada de un applet o su destrucción pasa por ejecutar los métodos stop() y destroy() del propio applet.

Ahora bien, estos métodos están concebidos para controlar la creación y destrucción del applet, pero de su ejecución se debe encargar el programador y esto es posible debido a la implementación de la interaz Runnable para la creación de hilos, por lo que nuestro applet internamente lo que estará haciendo es ejecutar un thread. Gracias a esta posibilidad, los applets pueden realizar cualquier tarea como una aplicación normal, aunque también destacar que están definidas ciertas limitaciones para algunas tareas.

UNOS EJEMPLOS: EL RELOJ Y LA MÚSICA

Vamos a desarrollar un sencillo ejemplo para ver cómo los applets ejecutan internamente threads que se encargan de la ejecución de tareas. El ejemplo del Listado 3 muestra un reloj en el navegador.

Como se observa en el código explicado, hemos declarado un thread denominado hilo que se encargará de mostrar el reloj en una página del navegador. La ejecución del applet comienza por el método init() que muestra en la parte inferior del navegador el mensaje Preparando Reloj ... , indicando así el inicio de la ejecución del thread. A continuación se ejecuta el método start() y es ahí donde el thread es creado para que comience su ejecución. A continuación el método se llama al método run() y a partir de este momento la ejecución se desarrolla de forma análoga a un "normal". Por este motivo, tiene que dejar tiempo de ejecución a los otros posibles threads, por lo que se duerme durante un periodo de tiempo tras el

Listado 3. Ejemplo del reloj en el navegador.

```
import java.awt.* ;
import java.applet.* ;
import java.util.Date;

public class reloj extends Applet implements Runnable{

   int horas, minutos, segundos;
   String mensaje ;
   Thread hilo = null;
   public void init (){
       showStatus ( "Preparando Reloj ... ");
   }
   public void start()        {
       if ( hilo == null )            {
              hilo = new Thread(this);
              hilo.start();
       }
   }
   public void run (){
       setBackground ( Color.white );
       setForeground ( Color.black );
       while ( hilo != null )         {
              try                     {
                     hilo.sleep (100);
              }
              catch ( InterruptedException e ){
                     /* Nada */;
              }

              Date  hora = new Date();
              horas = hora.getHours();
              minutos = hora.getMinutes();
              segundos = hora.getSeconds();
              if ( segundos < 10 )
                     mensaje = horas + " : " + minutos + " : 0"+segundos ;
              else
                     mensaje = horas + " : " + minutos + " : " + segundos ;
              repaint();
       }
   }
   public void paint( Graphics g ){
       g.drawString ( mensaje,10,10 );
   }
   public void stop()        {
       if ( hilo != null ) {
              hilo.stop();
              hilo = null ;
       }
   }
}
```

cual ejecuta la tarea de mostrar el reloj. Para conseguir los valores correctos recoge la hora del sistema y la presenta en pantalla. La visualización en los applets siempre es realizada por el método repaint(), en el sentido de que es el único método que puede realizar a llamada al método paint() que es el que verdaderamente dibuja la salida que visualizaremos en el navegador, en este caso, el reloj.

Como se ha descrito, la finalización de un applet se realiza en el método stop(), que en caso de que tengamos threads será utilizado para finalizarlos. En este caso, en este método llama al proceso de stop() del thread.

Si bien se observa que generar applet basándonos en la creación de threads es algo relativamente sencillo, la creación de applets que contengan ciertas características obliga a realizar ciertas reflexiones previas.

Por ejemplo, si un applet tiene que tocar una canción y el tamaño del fichero de música es algo grande, nuestro applet estaría detenido hasta que se finalizara la carga del fichero de música, lo cual aparentemente no es algo muy recomendable. Una solución simple para este tipo de problemas es crear un thread que se encargue de todas esas tareas. En el caso de la música, tendríamos que generar un thread que se encargara de cargar la música y tocarla.

El ejemplo del Listado 4 muestra la forma en que un thread realiza la tarea de fondo de cargar la música y ejecutarla, mientras que en el applet otro thread ejecuta la tarea de mostrar el reloj, por lo que el thread que muestra el reloj no espera a que la música se cargue y empiece a tocar.

La parte fundamental del código se encuentra en la clase Player, que es la encargada de recibir la referencia del applet mediante el cual puede acceder al método play() que se

Listado 4. Carga de la música y ejecución.

```
import java.awt.* ;
import java.applet.* ;
import java.util.Date;

public class musica extends Applet implements Runnable
{
Thread hilo = null;
Thread musica = null ;

int horas, minutos, segundos;
String mensaje ;

public void start()  {
  if ( hilo == null ) {
      hilo = new Thread(this);
      hilo.start();
      musica = new Thread ( new Player(this));
      musica.start();
  }
}

public void run ()  {
  setBackground ( Color.white );
  setForeground ( Color.black );

  while ( hilo != null ){
     try   {
           hilo.sleep (100);
           }
     catch ( InterruptedException e )   {
           ;
           }

     Date  hora = new Date();
     horas = hora.getHours();
     minutos = hora.getMinutes();
     segundos = hora.getSeconds();
     if ( segundos < 10 )
           mensaje = horas + " : " + minutos + " : 0"+segundos ;
     else
           mensaje = horas + " : " + minutos + " : " + segundos ;
     repaint();
     }
}

public void paint ( Graphics g )   {
  g.drawString ( mensaje,10,10 );
  }
```

Listado 4. (Continuación).

```
public void stop()   {
  if ( hilo != null )          {
       hilo.stop();
       hilo = null ;
       }
  if ( musica!= null )         {
       musica.stop();
       musica = null ;
       }
  }
}

class Player implements Runnable {
   private Applet m_Applet;

   Player ( Applet p_Applet )        {
        m_Applet = p_Applet ;
   }

   public void run (){
        m_Applet.play ( m_Applet.getCodeBase(),"sonido.au");
   }
}
```

encarga de cargar la música y tocarla. De este modo, tenemos dos threads ejecutándose a la vez, mientras que uno toca la música otro visualiza la hora en el navegador.

UTILIZACIÓN DE LOS THREADS EN LAS APLICACIONES CLIENTE/SERVIDOR

El uso de threads en los servidores de aplicaciones cliente/servidor permite que las peticiones de los clientes se atiendan a mayor velocidad, además de producir una menor carga en el sistema debido a que el trasiego de los threads es mucho menos que el producido por los procesos tradicionales que se crean en el servidor. Estas dos cualidades hacen ideales a los threads en el desarrollo de servidores en aplicaciones cliente/servidor.

UN SERVIDOR BÁSICO

Veamos un ejemplo de un servidor multithread básico. El código del Listado 5 es un servidor de eco que acepta múltiples conexiones.

La clase servidor es una clase que genera hilos por cada conexión que se efectúe. Dentro de un bucle que efectúa la llamada accept() se genera un thread que se encarga de gestionar la conexión, generando su stream de entrada y salida del socket cliente. Todo lo que recibe el thread servidor es reenviado hasta que el cliente desconecte, por lo que el servidor ejecuta el método cerrar_conexión() en el que se cierra el socket y se elimina el thread de la conexión actual.

Los applets son programas Java que requieren para su funcionamiento el uso de las técnicas de los threads

Una variante, ya demasiado afinada, consiste en generar un thread por cada stream del socket, esto es, se genera un thread para la gestión de los datos recibos y otro para los datos que se desea enviar al cliente, de forma que entre estos threads se establece un mecanismo de comunicación. Se trata de un buen ejercicio que realizar.

LA TÉCNICA THREAD POOL PARA SERVIDORES

De todos modos, este sistema se puede mejorar. Como se ha mencionado, la operación de generar un thread, si bien es mucho más económica que la de un proceso, es aún considerable. Además, la operación de eliminar también requiere su tiempo.

Por tanto, una solución a la que deberíamos llegar, podría pasar por generar un conjunto de threads que estén esperando las conexiones y realizar la gestión sobre ellos, de forma que una cantidad de thread se generan según se inicia el servidor y no se destruyen sino que se utilizan para distintas conexiones, con lo que se

Listado 5. Servidor de eco que acepta múltiples conexiones.

```
import java.net.*;
import java.io.*;
class servidor extends Thread{
Socket cliente ;
String cadena ;
servidor ( Socket conex ){
   cliente = conex ;
   }
public static void main ( String args[] ){
   ServerSocket serv ;
   Socket cli_socket;
   try {
        serv = new ServerSocket (7000);
        System.out.println ("Conexion en puerto 7000...");
        while ( true ) {
              cli_socket = serv.accept();
              System.out.println ("Una conexion ... ");
              new servidor( cli_socket).start();
        }
   }
   catch ( Exception e ) { ; }
}
public void run()    {
   try {
        DataInputStream  is =
                    new DataInputStream ( cliente.getInputStream());
        DataOutputStream os =
                    new DataOutputStream ( cliente.getOutputStream());
        while ( true ) {
              cadena = is.readLine();
              os.writeBytes (cadena );
              os.flush();
        }

   }
   catch ( Exception e ) {
        this.cerrar_conexion();
   }
}
public void cerrar_conexion (){
   System.out.println ("se cerro la conexion..");
   try  {
        if ( cliente != null )
              cliente.close();
        if ( this.isAlive() );
              this.stop();
   }
   catch ( Exception e )     { ; }
}
}
```

evita el trasiego de creación y destrucción de los threads.

La idea anterior es conocida como la técnica de thread Pool. Aplicada al ejemplo anterior tendríamos el código que aparece en el Listado 6.

Con el uso de multihilos siempre surge la necesidad de realizar operaciones que afecten a todos los threads

Estudiando el código detenidamente se puede observar que se inicia la creación de los threads de forma global, de tal forma que se crea una cantidad de hilos.

Estos threads son los encargados de realizar las conexiones con los clientes, pero ahora no tenemos un único punto donde esperamos conexiones, sino que hay tantos métodos accept() como threads generados. Otra ventaja es que los threads ya se han generado y no se destruyen, por lo que todo el tiempo que consume estas operaciones se ahorra.

Gracias a la utilización de los threads, podemos tener un applets que efectue varias tareas simultaneamente

Los problemas que puede plantear esta técnica se refieren a la dificultad que supone determinar el número óptimo de conexiones que realizar, si bien, siempre se puede implementar un método que avise que se ha alcanzado el número máximo de conexiones ocupadas o encolar las conexiones hasta que se puedan atender.

Listado 6. Técnica de thread Pool.

```
import java.net.*;
import java.io.*;
class servidor extends Thread{
private String cadena ;
ServerSocket servidor ;
Socket cliente;
static int num_conex = 20 ;
servidor ( ServerSocket serv_conex ){
   servidor = serv_conex ;
}
public static void main ( String args[] ){
   ServerSocket serv ;
   try {
        serv = new ServerSocket (7000);
        for ( int cont = 0 ; cont < num_conex ; cont ++ )
              new servidor(serv).start();
        System.out.println ("Conexion en puerto 7000 ...");
   }
   catch ( Exception e ) { ; }
}
public void run(){

   while ( true ) {
        try {
              Socket cliente = servidor.accept();
              DataInputStream  is =
                    new DataInputStream ( cliente.getInputStream());
                    DataOutputStream os =
                    new DataOutputStream ( cliente.getOutputStream());
              while ( true ) {
                    cadena = is.readLine();
                    os.writeBytes (cadena );
                    os.flush();
              }

        }
        catch ( Exception e ) {
              this.cerrar_conexion();
        }
   }
}
public void cerrar_conexion ()      {
   try  {
        if ( cliente != null )
              cliente.close();
   }
   catch ( Exception e )      { ; }
}
}
```

CONCLUSIÓN

Los grupos de threads permiten tener un control y gestión de todos los threads que agrupan de una forma fácil y eficaz.

La aplicación de los threads en los navegadores nos permite la creación de nuevas aplicaciones sobre éstos de manera que un applet puede estar tocando una música mientras al mismo tiempo, visualiza una animación gráfica y a su vez se puede estar calculando cualquier serie de datos, todo ello de manera simultanea.

El uso de threads en aplicaciones del tipo cliente/servidor logra la mejora en el rendimiento de éstas

Además de todo esto, la utilización de threads en aplicaciones del tipo cliente/servidor nos permite a los usuarios que éstas se ejecuten de forma mucho más eficiente y se descarga a la máquina que contiene el servidor del cambio de contexto tan "pesado" que generan los procesos denominados tradicionales.

BIBLIOGRAFÍA

- "Using Java". Joe Weber. QUE.
- "Java in a nutshell". David Flanagan. O'Reilly& Associates Inc.
- "Thinking in Java". Bruce Eckel. Prentice Hall.
- "Exploring Java". Patrick Niemeyer, Joshua Peck. O'Reilly & Associates Inc.

SVGALIB (y IV): El último escollo, manejo de periféricos

Jorge F. Delgado Mendoza (ernar@convex.es)

La gran mayoría de las aplicaciones informáticas, incluidos los juegos, responden a un modelo muy humano: el paradigma estímulo/respuesta. A lo largo de tres artículos hemos aprendido a responder... ha llegado el momento de recibir información desde el mundo exterior.

A lo largo de tres artículos hemos aprendido, paso a paso, los detalles relativos al funcionamiento de una de las librerías más interesantes de *Linux*: SVGALIB. La calidad de esta herramienta de programación gráfica ha permitido la aparición en *Linux* de programas tan interesantes como el *DOOM I* y *II* de *Id. Software*, así como otros muchos juegos entre los que destacan: *Abuse*, *Maelstrom*, etc.

En la primera entrega, concebida como una introducción a la API de esta librería, dividimos el artículo en tres partes. Comenzamos con una breve reseña teórica relativa a la arquitectura y el funcionamiento de SVGALIB. En ella aprovechamos para familiarizar al lector con la terminología que se iba a emplear a lo largo de la serie de artículos, de tal manera que palabras como *bpp*, *VGA*, *modos*, *resolución*, etc, no constituyeran una barrera para la comprensión de los diferentes conceptos.

A continuación describimos el procedimiento de instalación, detallando los diferentes pasos que debíamos dar para que la librería funcionara correctamente en nuestro sistema.

Para concluir, entramos a describir las funciones básicas de inicialización, así como de interrogación de las capacidades que los diferentes *drivers* ofrecen. No todas las tarjetas nos permiten realizar las mismas funciones, debido a lo cual el programador debe saber cuales son sus posibilidades en cada momento.

El segundo artículo comenzó con una descripción más detallada de algunos conceptos fundamentales, para a continuación describir las primitivas más importantes asociadas a la propia SVGALIB. En él aprendimos a dibujar círculos y líneas, a rellenar rectángulos con un color determinado, a manejar la memoria de vídeo, etc.

El tercero completaba las ideas expuestas en los dos anteriores con la descripción del manejo de VGAGL, una librería de más alto nivel construida sobre SVGALIB. Aprovechando la circunstancia, introdujimos conceptos tan interesantes como *page flipping*, pantallas virtuales, *clipping*, etc, describiendo las diferentes primitivas que la librería ofrece para poder trabajar con ellos.

Ahora el lector se podría preguntar: ¿Qué más puede quedar por aprender que justifique una nueva entrega de la serie?. Y la verdad es que es una buena pregunta. Podríamos seguir desgranando las restantes funciones gráficas, sin embargo ninguna de ellas aporta nada nuevo y su utilización debería

ser evidente después de estudiar los ejemplos de los restantes artículos.

Para poder responder satisfactoriamente a esta pregunta deberíamos formularnos otra: ¿Qué forman un montón de gráficos, - y quizá sonido, pero esa es otra historia, - paseando a sus anchas por la pantalla de nuestro ordenador?

Está claro que, lo que sea, no es un juego. Quizá podríamos llamarlo una demo, pero definitivamente no es un juego. ¿Qué es lo que le falta?. La respuesta es evidente: **interacción.** Un juego necesita un jugador, alguien que se comunique con él, que reciba estímulos y responda a los mismos. Sin embargo, nada de lo que hasta ahora hemos aprendido nos permite dar una respuesta externa al ordenador.

¡Ya hemos encontrado la respuesta a nuestra pregunta!. Para que la serie de artículos esté completa, debemos proporcionar al lector alguna manera de comunicarse con el ordenador. Por lo tanto, la última entrega de la serie de artículos sobre SVGALIB se va a dedicar íntegramente al manejo de periféricos de entrada.

Un análisis exhaustivo de todos los artilugios que podemos encontrar en el mercado, para comunicarnos con nuestro PC, se saldría completamente del objetivo de la serie, - además de necesitar unos cuantos centenares de páginas. - Por lo tanto, a lo largo de este artículo vamos a describir el funcionamiento de los dos más importantes: el ratón y el teclado.

REQUISITOS

CONOCIMIENTOS

Como ya viene siendo costumbre, es necesario un conocimiento relativamente profundo del lenguaje de programación C para poder entender todos y cada uno de los detalles que se exponen en el artículo.

La razón es muy simple, todos los ejemplo de la serie han sido realizados en este lenguaje y los puntos de entrada a la API están diseñado para su inclusión en programas desarrollados en este lenguaje.

Aunque para compilar los diferentes ejemplos no es necesario, cualquier intento de realizar aplicaciones serias utilizando la librería SVGALIB implicará dominar el manejo de la herramienta *make*, así como un conocimiento detallado de las posibilidades del compilador .

Finalmente, y aunque no es estrictamente necesario, el diseño modular del curso sugiere el acceso a los restantes artículos de la serie para así poder seguir de una manera cómoda y directa los diferentes conceptos que se van a describir a continuación.

DOCUMENTACIÓN

Sería interesante para el lector tener acceso a los demás artículos de la serie, aparecidos en los números 47 a 49 de la revista. Sin embargo, para todo aquél que desee aumentar sus conocimientos en este campo, incluimos dos referencias bibliográficas obligadas:

- La primera de ellas es lo que se denomina la Biblia de la programación de tarjetas gráficas. Con varias ediciones a sus espaldas, en las que el autor ha expandido y completado la información relativa a las diferentes tarjetas gráficas, el libro titulado: *Programmer's Guide to SVGA Cards* publicado por *Addison Wesley*, - cuyo autor es *Richard Ferraro*, - es probablemente la mejor fuente de información disponible.

- Por otro lado, si lo que deseamos es obtener información acerca de programación gráfica en general, las mejores fuentes están *online* en *Internet*. Un buen punto de arranque para obtener esta información se puede obtener apuntando nuestro explorador favorito hacia:

  http://www.cs.umu.se/~christer /GR/developement.html

Finalmente, todas las primitivas de la API de SVGALIB están documentadas en las páginas de manual correspondientes. Mediante la orden:

```
man svgalib
```

Podremos acceder a una descripción completa de la librería, así como a los nombres de las páginas de manual de todas sus primitivas.

SOFTWARE

Es imprescindible tener instaladas, al menos, las siguientes herramientas en nuestro sistema:

- El compilador de lenguaje C de *GNU:* GCC.
- La utilidad *make*.
- Una versión de la librería gráfica posterior a la 1.2.10.

Todos los ejemplos del artículo se han compilado en la distribución 5.0 de *RedHat*, la cual incluye las siguientes versiones del *software*:

- GCC versión 2.7.2.3
- GNU make versión 3.76.1

Además, la librería gráfica utilizada para todos y cada uno de los ejemplos es SVGALIB versión 1.2.13.

HARDWARE

Aunque cualquier tarjeta compatible VGA se puede utilizar con SVGALIB, cuanto mejor soporte tengamos

para ella más posibilidades tendremos de conseguir utilizar modos de alta resolución y una enorme multitud de colores.

Para comprobar si nuestra tarjeta está soportada, es suficiente consultar la página de manual genérica de SVGALIB. En la sección 4ª: *"OVERVIEW OF SUPPORTED SVGA CHIPSETS AND MODES"*, podremos encontrar una lista con toda la información relativa a los modos soportados por los *chipsets*.

En nuestro caso, todos los programas de demostración de los diferentes artículos se han probado con una *S3D ViRGE* con 2 Mbytes de memoria de vídeo. El *driver* incluido en SVGALIB 1.2.13 permite conseguir resoluciones de hasta 1024x768 en 256 y 64K colores, así como de 800x600 en modos de 24 y 32 *bits* por punto.

EL RATÓN

Es probablemente el dispositivo más utilizado, después del teclado. Dado que la mayoría de las aplicaciones modernas, sean lúdicas o no, utilizan de alguna manera este periférico, los creadores de SVGALIB decidieron introducir soporte para el mismo.A continuación describimos la utilización de las funciones más relevantes que ofrece la librería.

FUNCIÓN VGA_GETMOUSETYPE

Esta función permite al programador acceder al tipo de ratón que se ha configurado para la librería. El valor del mismo se obtiene del fichero:

```
/etc/vga/libvga.config
```

Los distintos tipos soportados incluyen:

- *Microsoft*
- *Logitech*
- *MouseSystems*
- *MMSeries*
- *BusMouse*
- *PS/2*
- *Logiman*
- *Spaceball* (Ratón 3d *Spacetec Spaceball).*

Además, es posible utilizar directamente el demonio *gpm* para acceder a la información del ratón. En este caso, el tipo de ratón que obtenemos es artificial, - no corresponde a ningún protocolo concreto, - y se denomina:

```
GPM
```

Una llamada a la función devuelve un identificador de tipo entero para cada tipo de ratón. Estos valores están definidos en los ficheros de cabecera de la librería. Podemos consultar sus nombres en las páginas de manual de la función.

FUNCIÓN VGA_SETMOUSESUPPORT

Esta función deberá ser invocada antes de utilizar ningún tipo de primitiva relacionada con el ratón. Se limita a activar el soporte para ratón, - utilizando un "1" como argumento, - o desactivarlo si el argumento es un cero.

FUNCIÓN MOUSE_UPDATE

Esta función accede al *hardware* del periférico, actualizando las variables internas de la librería con los datos que se encuentre. Si encuentra algún tipo de actividad, devuelve un valor positivo. Si el ratón no se ha movido, se devuelve un cero.

FUNCIÓN MOUSE_WAITFORUPDATE

Pueden existir casos en los que el desarrollador desee bloquear la ejecución de un programa hasta que se produzca un determinado evento en el puntero. Aunque la función descrita anteriormente, - *mouse_update,* - permite realizar una espera activa, en una máquina multitarea es mucho más conveniente dormir al proceso hasta que se produzca ese evento.

La rutina *mouse_waitforupdate* cumple exactamente esa misión. Permanece bloqueada hasta que se reciben datos desde el ratón. En ese momento, actualiza las variables internas y devuelve el control al programa.

FUNCIÓN MOUSE_SETWRAP

Una de las decisiones con las que se enfrenta un diseñador de interfaces gráficas, es el hecho de especificar el comportamiento del cursor cuando éste alcanza el límite de la pantalla.

SVGALIB nos permite especificar, para cada coordenada, si deseamos que el ratón que desaparezca por un lado aparezca por el otro o bien que permanezca en el límite de la pantalla.

Esta función se invoca con un parámetro que puede ser:

- *MOUSE_WRAP:* el ratón aparece por un lado al desaparecer por el otro.
- *MOUSE_NOWRAP:* el ratón permanece inscrito en los límites de la pantalla.

Adicionalmente, es posible controlar individualmente hasta 6 ejes de traslación. Para ello podemos substituir el parámetro anterior por una máscara creada mediante la operación lógica *'or'* de los siguientes valores:

```
MOUSE_WRAPX
MOUSE_WRAPY
MOUSE_WRAPZ
MOUSE_WRAPRX
```

MOUSE_WRAPRY
MOUSE_WRAPRZ

Normalmente nosotros utilizaremos los valores genéricos o, en algún caso especial, los que controlan las coordenadas *'x'* e *'y'* individualmente.

FUNCIONES MOUSE_GETX Y MOUSE_GETY

Estas funciones obtienen los valores de las dos coordenadas espaciales que determinan la posición del puntero en la pantalla.

Al ser invocadas, devuelven un valor entero que se corresponde con la coordenada deseada. Al igual que *mouse_getbutton*, estas funciones acceden a las variables internas debido a lo cual es necesario utilizar alguna de las primitivas descritas anteriormente cuya misión es la de actualizar el contenido de esas variables.

FUNCIÓN MOUSE_GETBUTTON

De manera similar a las primitivas anteriores, esta función devuelve información relativa a la activación de los botones del puntero. Es posible seleccionar individualmente el botón cuya activación se desea comprobar, realizando un *and* lógico a nivel de bit del valor devuelto.

La librería facilita una serie de máscaras para simplificar la tarea. Estas máscaras son:

MOUSE_LEFTBUTTON
MOUSE_MIDDLEBUTTON
MOUSE_RIGHTBUTTON
MOUSE_FOURTHBUTTON
MOUSE_FIFTHBUTTON
MOUSE_SIXTHBUTTON
MOUSE_RESETBUTTON

Si la función devuelve un valor nulo, ninguno de los botones ha sido pulsado. De manera análoga a las rutinas relativas al cáculo de coordenadas, es necesario invocar las primitivas de actualización antes de utilizar esta función.

FUNCIÓN MOUSE_SETPOSITION

Aunque la mayoría de las funciones asociadas con el puntero permiten obtener sus valores actuales, esta función permite forzar una posición determinada desde dentro de un programa.

De esta manera, una llamada como la siguiente:

```
mouse_setposition ( 300 , 740 );
```

Coloca el cursor en la posición cuya abscisa es 300, siendo su ordenada 740.

EL TECLADO

Heredado de las máquinas de escribir, el teclado es sin duda el dispositivo de entrada/salida por excelencia. Es posible pasarse sin disco, sin ratón, e incluso sin disco flexible. Sin embargo, es casi imposible utilizar un ordenador sin un teclado de algún tipo.

MODOS DEL TECLADO: RAW VS. NORMAL

SVGALIB nos permite utilizar el teclado de dos maneras diferentes, según el grado de detalle que queramos tener. El modo *normal*, permite acceder a los caracteres ASCII pulsados, una vez han sido procesados por el *kernel* y los sistemas de gestión de las distintas disposiciones.

Sin embargo, es posible que un programa necesite acceder a una tecla en concreto, o bien enlazar una tecla determinada con un comportamiento diferente al que tiene. En estos casos, no queda más remedio que acceder al teclado a más baso nivel, en lo que se denomina modo *raw*.

Todas las funciones que comienzan por el prefijo *keyboard_*, pertenecen al modo *raw*, mientras que las que comienzan por *vga_* pertenecen al modo *normal*. A continuación describiremos las más importantes.

FUNCIÓN KEYBOARD_INIT

Esta primitiva inicializa el modo *raw* del teclado, siendo necesario invocarla antes de realizar ningún tipo de operación.

Esta función devuelve 0 en caso de haber tenido éxito y -1 en caso contrario.

Una vez activado este modo, la librería nos permite aislar cada pulsación de una tecla de manera independiente. Además, tenemos la posibilidad de redefinir los códigos que se asignan a cada tecla.

En cualquier caso, los códigos que nos permiten cambiar de consola virtual o el de interrupción, permanecen inalterables.

FUNCIÓN KEYBOARD_CLOSE

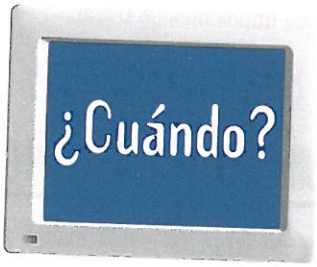
Keyboard_close desactiva el modo *raw* del teclado, permitiendo utilizar las funciones estándar de C, así como las primitivas del modo *normal*.

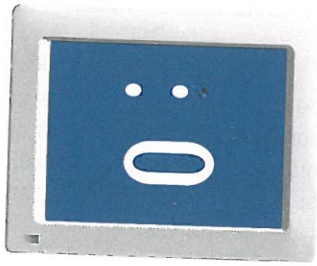
Es **imprescindible** llamar a esta función antes de terminar el programa, ya que se pueden producir efectos no deseados si no lo hacemos.

FUNCIÓN KEYBOARD_UPDATE

De manera similar a las funciones de actualización del ratón, esta primiti

INTERNET ASEQUIBLE





va recorre el teclado para ver si se ha pulsado alguna tecla. También este caso existe una llamada bloqueante, además de la versión no bloqueante que se muestra aquí.

Si no se ha pulsado ninguna tecla, la función devuelve cero. En caso contrario se devuelve un valor positivo asociado con el código de la tecla o teclas pulsadas.

FUNCIÓN KEYBOARD_KEYPRESSED

Esta función permite saber si una tecla, cuyo código le pasamos como argumento, ha sido pulsada. De manera idéntica al caso del ratón, es necesario activar la primitiva de actualización antes de llamar a esta función.

Para acceder a la tabla con los distintos códigos de tecla, podemos consultar los ficheros de cabecera, así como la página de manual correspondiente.

Esta función se encarga de devolver *KEY_PRESSED* si la tecla ha sido pulsada o *KEY_NOTPRESSED* en caso contrario.

FUNCIÓN KEYBOARD_GETSTATE

Esta función es similar a la anterior. Su misión es acceder al teclado y comprobar que teclas se han pulsado. *Keyboard_getstate* devuelve un array de valores cuyo índice es el código de cada tecla, siendo su contenido la información relativa al estado: pulsada o levantada. También es necesario realizar actualizaciones de los valores internos antes de realizar esta llamada.

FUNCIÓN KEYBOARD_CLEARSTATE

Esta función, la última que vamos a ver del modo *raw*, reinicializa el valor de las variables internas de la librería, de tal manera que ninguna tecla esté pulsada.

Una llamada a *getstate* colocada justo detrás de *clearstate* devolverá siempre *KEY_NOTPRESSED*.

FUNCIÓN VGA_GETKEY

Similar al típico *INKEY$* de BASIC, esta llamada explora el teclado para ver si se ha pulsado alguna tecla. En caso afirmativo, devuelve código ASCII de la misma. En caso contrario devuelve cero.

Esta función retorna instantáneamente, debido a lo cual su utilización no es recomendable en rutinas de espera.

FUNCIÓN VGA_GETCH

Similar a la anterior, esta función analiza la cola de entrada del teclado para ver si se ha pulsado alguna tecla. A diferencia de *getkey*, esta primitiva bloquea la ejecución del programa hasta que se pulsa alguna.

En ningún caso deberemos mezclar funciones de los diferentes modos del teclado. Los resultados son impredecibles y pueden dar lugar a errores casi imposibles de trazar.

CONCLUSIONES

A lo largo de estos cuatro artículos hemos ido aprendido paso a paso a manejar la librería gráfica SVGA para *Linux*, llamada SVGALIB.

Para ello hemos ido describiendo poco a poco todos aquellos pasos que son necesarios dar para crear un programa que utilice, en profundidad, las posibilidades que ofrecen las tarjetas de vídeo de los PC's.

En una primera entrega, eminentemente teórica, asimilamos el procedimiento de instalación y describimos los conceptos básicos que todo desarrollador debería tener presentes a la hora de crear un programa basado en SVGALIB. Además introdujimos al lector en el manejo de las primitivas básicas necesarias para empezar a crear imágenes en la pantalla.

En el segundo artículo abordamos conceptos relativos al *hardware* , el cual condiciona las posibilidades que nos va a ofrecer la librería. Así describimos los diferentes modos de acceso a la memoria de vídeo, aprendimos a manejar una paleta de colores, etc.

La anterior entrega del curso se ciñó a la descripción de la API de programación de VGAGL, un subconjunto de SVGALIB destinado a virtualizar el acceso al *hardware*. Mediante su utilización descubrimos un nuevo mundo de posibilidades: *doble buffering*, *page flipping*, etc. Además aprendimos a realizar operaciones gráficas más complejas, las cuales incluían: zonas de corte, creación de figuras geométricas, copias de un lado a otro de la pantalla, etc.

En el artículo que cierra la serie, nos hemos ceñido a algo que no es el manejo de gráficos propiamente dicho. Sin embargo, las aplicaciones para las que está orientada SVGALIB utilizan, en la gran mayoría de los casos, dispositivos de entrada/salida. Al ser el teclado y el ratón los más importantes, hemos considerado oportuno proporcionar al lector información suficiente como para que sea capaz de integrarlos en sus aplicaciones.

[1] En el segundo artículo de esta serie, titulado *"Profundizando en SVGALIB"* aparecido en el número 48 de esta misma revista, describimos por encima algunas de las opciones que nos facilitaba el compilador de lenguaje C de *GNU*, conocido como *GCC (Gnu C Compiler)*.

SÍ, ASEQUIBLE

NO ENCONTRARÁS UNA OFERTA MEJOR

¿Cómo?

Con un completo curso multimedia que te explicará desde en qué consiste el correo electrónico hasta qué es el FTP. Y, con conexión gratuita a Internet hasta el 31 de diciembre de 1.998.

¿Cuánto?

Por sólo 3.995 ptas. Iva incluido.

¿Dónde?

Podrás encontrarlo en librerías, quioscos, tiendas de informática, Vips, Crisol, Fnac, El Corte Inglés, Casa del libro o en el teléfono 91-6614211.

¿Cuándo?

Ya a la venta.

ABETO editorial

C/ Aragoneses, 7
28108 Pol. Ind. Alcobendas (MADRID)
Tel.: (91) 661 42 11* Fax: (91) 661 43 86
http://www.abetoed.es

También versión sin conexión por 2.995 ptas.

Programación básica en Windows (IV)

Jorge Figueroa (erde@arrakis.es)

El poder incorporar a nuestros propios programas recursos multimedia como gráficos, iconos, cursores, sonidos o el poder crear nuevas fuentes de letra es algo que se puede considerar indispensable actualmente. Por esta razón, en este capítulo, se va a mostrar la forma de poder hacerlo de manera sencilla.

Tras haber explicado en el anterior capítulo de este pequeño curso de programación la forma de poder usar las famosas barras de desplazamiento en nuestros programas, la presente entrega se va a dedicar a explicar la forma de poder incorporar diversos recursos de Windows a nuestras propias aplicaciones así como la forma de crear nuestros propios tipos de letra, de forma que estas puedan mostrar gráficos, iconos, nuevos cursores, reproducir sonidos, e incorporar textos con el aspecto que nosotros deseemos.

Crear una fuente es tan fácil como rellenar los campos de LOGFONTA y llamar a la función de creación

USO DE RECURSOS DE WINDOWS

Actualmente casi todos los programas que se diseñan en Windows incorporan diversos recursos como son iconos, gráficos y sonidos, lo cual da muchas más posibilidades a las aplicaciones a la vez que las hacen más interactivas.

Casi todos los compiladores de C/C++ que existen en el mercado en la actualidad disponen de herramientas integradas en el propio entorno de desarrollo (IDE) para que el programador que los utilice pueda crear, editar y guardar sus propios recursos Windows para poder así después añadirlos a sus propios programas.

Por otro lado, en los casos en que un editor específico no esté disponible en el propio IDE, normalmente suele ser debido a que el propio sistema operativo ya proporciona alguno como es por ejemplo la grabadora de sonidos de Windows, que permite generar ficheros wav, o porque son en sí mismos muy complejos de crear como son por ejemplo los ficheros de música o las películas AVI, las cuales necesitan de software especializado e incluso hardware especial (tarjeta de sonido, digitalizadora de TV, etc.)

Por ahora nos vamos a encargar de mostrar solamente la forma de poder incorporar algunos recursos muy sencillos como puedan ser un icono, qué será un fichero con extensión .ico (myicon.ico), un cursor (mycur.cur), un pequeño gráfico (mybmp.bmp) y un fichero de sonido en formato .WAV (wav.wav).

EL PROGRAMA

En el Listado 1 se ha incluido un programa detallado que realiza la función deseada: mostrar los recursos explicados por pantalla y reproducir el sonido.

Antes de copiar el programa y poder ejecutarlo, el lector deberá conseguir unos ficheros de las extensiones mencionadas (.ico, .cur, .bmp y .wav) para poder usarlos como ficheros a cargar en el ejemplo. En el caso de que no encuentre ficheros de estos tipos, el propio lector deberá crear los suyos propios.

Otro dato muy importante es que este programa, debido a las funciones que usa, necesita para poder ser compilado y convertido en un ejecutable de una librería llamada *<winmm.lib>* (usado por el *linker*).

Por último, decir que para crear un programa que use recursos es necesario crear también un archivo de recursos (*recurso.rc* en este caso), el cual se explicará más adelante.

EL CÓDIGO

Como se puede observar, el código es necesario para poder mostrar los recursos del ejemplo es relativamente corto y sencillo. A continuación se detallan todas las líneas.

EL ICONO

Incorporar un icono personalizado a un programa implica dos pasos. Primero de todo se debe asignar a *szIconName* el nombre de icono definido:

```
IconName[]= "NewIcon";
```

Listado 1. Ejemplo de uso de recursos en un programa.

```
#include <windows.h>
#include <mmsystem.h>
LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);
char szProgName[] = "Progname";
char szIconName[] = "NewIcon";
char szCursorName[]= "NewCursor";
char szBMPName[] = "BMImage";
HBITMAP hbitmap;

int WINAPI WinMain(HINSTANCE hInst, HINSTANCE hPreInst, LPSTR lpszCmdline, int nCmdShow)
{
 HWND hWnd;
 MSG lp;sg;
 WNDCLASS WcApp;

 hBitmap = LoadBitmap(hInst, szBMPName);

 wcApp.lpszClassName  = szProgName;
 wcApp.hInstance      = hInst;
 wcApp.lpfnWndProc    = WndProc;
 wcApp.hCursor        = LoadCursor(hInst,szCursorName);
 wcApp.hIcon          = LoadIcon(hInst, szIconName);
 wcApp.lpszMenuName = NULL;
 wcApp.hbrBackground = GetStockObject(WHITE_BRUSH);
 wcApp.style          = CS_HREDRAW | CS_VREDRAW;
 wcApp.cbClsExtra     = 0;
 wcApp.cbWndExtra     = 0;
 wcApp.cbSize         = sizeof(WNDCLASSEX);
 wcApp.hIconSm        = LoadIcon(NULL, IDIASTERISK);
 If(!RegisterClass (&wcApp))
  Return 0;

hWnd= CreateWindow(szProgName, "Ejemplo 1. Uso de recursos en Windows",
WS_OVERLAPPEDWINDOW, CW_USEDEFAULT, CW_USEDEFAULT, CW_USEDEFAULT, CW_USEDEFAULT, (HWND) NULL,  (HMENU) NULL, (HANDLE) hInst, (LPSTR) NULL);

 ShowWindow(hWnd, nCmDShow);
 UpdateWindow(hWnd);
 While (GetMessage(&lpMsg, NULL, 0, 0))
    {
      TranslateMessage(&lpMsg);
      DispatchMessage(&lpMsg);
    }
 return(lpMsg.wParam);
}

LRESULT CALLBACK WndProc(HWND hWnd, UINT messg, WPARAM wParam,
LPARAM lParam)
{
 HDC hdc;
 HDC hmdc;
 PAINTSTRUCT ps;
 BITMAP bm;
```

Listado 1. (Continuación).

```
switch(messg)
  case WM_PAINT:
     hdc = beginPaint(hWnd, &ps);

     hdmc = CreateCompatibleDC(hdc);
     SelectObject(hdmc, hBitmap);
     GetObject(hBitmap, sizeof(bm), (LPSTR), &bm);
     BitBlt(hdc, 75, 75, bm.bmWidth, bm.bmHeigh, hdmc, 0, 0, SRCCOPY);
     sndPlaySound("WAV:WAV",SND_ASYNC);
     DeleteDC(hdmc);

     ValidateRect(hWnd, NULL);
     EndPaint(hWnd_NULL);
     break;

  case WM_DESTROY:
     DeleteObject(hBitmap);
     PostQuitMessage(0);
     break;

  default:
     return(DefWindowProc(hWnd,messg,wParam,lParam));
  }
  return(0);

  }
```

Una vez realizado esto se debe hacer que el programa lo asigne a la clase de ventana:

```
wcApp.hIcon = LoadIcon(hInst, szIconName);
```

Además de lo explicado, resaltar también que el programa usa una versión extendida de la estructura de clase de Windows para poder así incluir un pequeño icono (los famosos iconos reducidos de Windows 95). El código correspondiente es el que describimos a continuación:

```
wcApp.lpszClassName=szProgName;
+
wcApp.hCursor = LoadCursor(hInst, szCursor-
      Name);
wcApp.hIcon = LoadIcon(hInst, szIconName);
+
+
+
wcApp.hIconSm = LoadIcon (NULL,
      IDI_ASTERIK); // Icono pequeño
if( !RegisterClass (&wcApp)) return 0;
```

Con el código mostrado, lo que windows hace es registrar la clase extendida como una llamada a la función *RegisterClassEx( );*. Windows usará un icono pequeño, identificado por *wcApp.hIconSm* en vez del icono grande cargado antes.

Para no complicar más el programa, se ha usado uno de los iconos existentes por defecto.

EL CURSOR

Añadir el cursor como se puede ver en el código es tan simple como el hecho de añadir un icono. A *szCursorName* se le debe asignar el nombre del cursor que hemos definido para el fichero de cursor:

```
szCursorname[]= "NewCursor";
```

Una vez hecho esto, el programa debe asignar el cursor definido a la clase de ventana (igual que se hacía con el icono):

```
wcApp.hCursor = LoadCursor(hInst, szCursor-
      name);
```

Hecho todo esto, una vez sea creada la ventana, el cursor asociado a ella se activará automáticamente.

EL GRÁFICO

El gráfico es un elemento algo más complejo de incorporar, aunque no mucho más.

Primero de todo se debe asociar el nombre del gráfico mediante la siguiente línea:

```
hBitmap = LoadBitmap(hInst, szBMPName);
```

Después, antes de poder dibujarlo en pantalla, tenemos que el programa crea un contexto de dispositivo de memoria, lo que hará que el gráfico quede seleccionado:

```
hmdc= CreateCompatibleDC(hdc);
SelectObject(hmdc,hBitmap);
GetObject(hBitmap, sizeof(bm), (LPSTR),
      &bm);
BitBlt(hdc, 75, 75, bm.bmWidth, bm.bmHeight,
      hmdc, 0, 0, SRCCOPY);
```

La función CreateCompatibleDC se usa para crear un contexto de dispositivo de memoria que sea compatible con el contexto de dispositivo actual.

Después, con *SelectObject( )* el programa selecciona el mapa de bits identificado pro hBitmap dentro del contexto de dispositivo de memoria y con la función *GetObject( )* se obtiene el ancho y alto del gráfico, datos que son necesarios para poder usar la función *BitBlt( )*, la cual copia el gráfico

desde el contexto de dispositivo de memoria a su contexto de dispositivo compatible.

EL SONIDO

Como se puede comprobar en el código, el sonido también es fácil de añadir a un programa. Para reproducir el sonido *wav.wav*, simplemente tenemos que realizar una llamada a la función sndPlaySound( ).

Como curiosidad, basta decir que esta función acepta trabajar de forma sincronizada o no. Si seleccionamos el modo asíncrono (SND_ASYNC), el control del programa se devuelve al procedimiento tan pronto la función empiece la reproducción del sonido, por lo que el programa podrá seguir su ejecución aún estando el sonido reproduciéndose.

EL ARCHIVO DE RECURSOS

El archivo de recursos es un archivo donde se indica al compilador información sobre los recursos usados por el programa que se quiere compilar. Este archivo se lo compila automáticamente el compilador durante la conversión a ejecutable. Una vez el archivo de recursos es compilado, este se almacena en disco como un fichero con extensión *.res*.

El archivo de recursos necesario para poder compilar el archivo de ejemplo del Listado 1 es el siguiente:

```
NEWICON     ICON     DISCARDABLE
        "myicon.ico"
NEWCURSOR CURSOR DISCARDABLE
        "mycur.cur"
BMIMAGE     BITMAP   MOVEABLE PURE
        "mybmp.bmp"
```

Listado 2. Lista de constantes usadas en las definiciones de fuentes.

```
#define FM_DONTCARE          0
#define FM_THIN              100
#define FM_EXTALIGHT         200
#define FM_LIGHT             300
#define FM_NORMAL            400
#define FM_MEDIUM            500
#define FM_SEMIBOLD          600
#define FM_BOLD              700
#define FM_EXTRABOLD         800
#define FM_HEAVY             900
#define FM_ULTRALIGHT        FM_EXTRALIGHT
#define FM_REGULAR           FM_NORMAL
#define FM_DEMIBOLD          FM_SEMIBOLD
#define FM_ULTRABOLD         FM_EXTRABOLD
#define FM_BLACK             FM_HEAVY

#define  ANSI_CHARSET        0
#define  UNICODE_CHARSET     1
#define  SYMBOL_CHARSET      2
#define  SHIFTJIS_CHARSET    128
#define  HANGEUL_CHARSET     129
#define  CHINESEBIG5_CHARSET 136
#define  OEM_CHARSET         255

#define  OUT_DEFAULT_PRECIS  0
#define  OUT_STRING_PRECIS   1
#define  OUT_CHARACTEC_PRECIS 2
#define  OUT_STROKE_PRECIS   3
#define  OUT_TT_PRECIS       4
#define  OUT_DEVICE_PRECIS   5
#define  OUT_RASTER_PRECIS   6
#define  OUT_TT_PRECIS       7
#define  OUT_OUTLINE_PRECIS  8

#define CLIP_DEFAULT_PRECIS  0
#define CLIP_CHARACTER_PRECIS 1
#define CLIP_STROKE_PRECIS   2
#define CLIP_MASK            0Xf
#define CLIP_LH_ANGLES       (1<<4)
#define CLIP_TT_ALWAYS       (2<<4)
#define CLIP_EMBEBED         (8<<4)

#define DEFAULT_QUALITY      0
#define  DRAFT_QUALITY       1
#define PROOF_QUALITY        2

#define DEFAULT_PITCH        0
#define FIXED_PITCH          1
#define VARIABLE_PITCH       2
#define FF_DONTCARE          (0<<4)
#define FF_ROMAN             (1<<4)
#define FF_SWISS             (2<<4)
#define FF_MODERN            (3<<4)
#define FF_SCRIPT            (4<<4)
#define FF_DECORATIVE        (5<<4)
```

Como se puede comprobar en las líneas del fichero explicado, la primera palabra de cada línea corresponde al nombre con el que se identifica cada uno de los recursos dentro del programa, de forma que no es necesario tener que trabajar dentro del programa con los nombres de fichero sino simplemente con el nombre identificativo que se le haya dado.

USO DE FUENTES EN WINDOWS

Windows posee una gran capacidad de control sobre los diversos tipos de fuentes que dispone instalados.

Con el fin de poder manipular las fuentes de forma sencilla, muchos de los parámetros programables de las mismas se definen en estructuras como son TEXTMETRIC, NEWTEXTMETRICA y LOGFONTA, las cuales son estándar y están declaradas en el fichero *wingdi.h*.

Al ser la cantidad de parámetros modificables en las estructuras bastante grande, Windows proporciona todos los valores como constantes, de forma que no sea necesario recordar todos los números asociados a cada atributo.

En el listado 2 puede ver la relación de constantes, las cuales se encuentran definidas en el fichero *wingdi.h*

LA ESTRUCTURA NEWTEXTMETRICA

Como se ha dicho anteriormente, la estructura TEXTMETRIC posee varios campos donde podemos indicar atributos de los definidos en el Listado 2 y con lo que el programador posee total libertad para poder definir fuentes a medida, tal y como él mismo decida.

Listado 2. (Continuación)

```
#define OEM_FIXED_FONT          10
#define ANSI_FIXED_FONT         11
#define ANSI_VAR_FONT           12
#define SYSTEM_FONT             13
#define DEVICE_DEFAULT_FONT     14
#define DEFAULT_PALETTE         15
#define SYSTEM_FIXED_FONT       16

#define TMPF_FIXED_PITCH        0x01
#define TMPF_VECTOR             0x02
#define TMPF_DEVICE             0x03
#define TMPF_TRUETYPE           0x04
```

La estructura NEWTEXTMETRICA posee una definición como la que sigue:

```
typedef struct tagTEXTMETRICA
 {
  LONG tmHeight;
  LONG tmAscent;
  LONG tmInternalLeading;
  LONG tmExternalLeading;
  LONG tmAveCharWidth;
  LONG tmMaxCharWidth;
  LONG tmWeight;
  LONG tmOverhang;
  LONG tmDigitizedAspecX;
  LONG tmDigitizedAspecY;
  LONG tmFisrtChar;
  LONG tmLastChar;
  LONG tmDefaultChar;
  LONG tmbreakChar;
  LONG tmItalic;
  LONG tmUnderlined;
  LONG tmStructOut;
  LONG tmPitchAndFamily;
  LONG tmCharSet;
  DWORD ntmFlags;
  UINT  ntmSizeEM;
  UINT ntmCellHeigh;
  UINT ntmAvgWidth;
 } NEWTEXTMETRICA;
```

Señalar que estos atributos sólo se aplican a fuentes de tipo Truetype.

De los parámetros indicados en esta estructura, los que más se usan para definir nuevos fuentes son los siguientes: *tmHeigh, tmAscent, tmDescent, tmInternalLeading, tmExternalLeading* y *tmAveCharWidth*.

LA ESTRUCTURA LOGFONTA

Además de la mencionada TEXTMETRICA, también hay otra estructura muy usada en programación de fuentes que es la siguiente:

```
typedef struct tagLOGFONTA
 {
  LONG lfHeight;
  LONG lfWidth;
  LONG lfEscapement;
  LONG lfOrientation;
  LONG lfWeigth;
  BYTE  lfItalic;
  BYTE  lfUnderline;
  BYTE  lfStrikeOut;
  BYTE  lfCharSet;
  BYTE  lfOutPrecision;
  BYTE  lfClipPrecision;
  BYTE  lfQuality;
```

Algunos Ratones son muy curiosos...

Demo disponibl
en nuestra Web

DadvinaFence
Seguridad en sus Aplicaciones.

Uno de los riesgos que sufren las aplicaciones informáticas es la posibilidad de su uso por personas no autorizadas.
DadvinaFence incorpora tres componentes OCX o VCL:**ApplicationLock**, **FormLock** y **NetMessenger**, proporcionando al programador un sistema de seguridad activo fácil de integrar y totalmente configurable.

- Versiones VCL para Delphi 3.0/2.0, C++ Builder 3 y OCX para Visual Basic 5.0.
- Asignación de permisos y privilegios sobre los objetos existentes en las pantallas de las aplicaciones.
- Autentificación de usuarios mediante el uso de identificación y contraseña.
- Detección de intentos fallidos continuados de entrada al sistema,avisando al instante a los administradores de seguridad de las aplicaciones.
- Centralización de los accesos (login) a las bases de datos de la aplicación.
- Registro de entradas y salidas.
- Sistema modular fácilmente actualizable en tiempo de ejecución y de diseño.

etc...

Sus clientes se lo Agradecerán

Sólo por 31.500* ptas.

Dadvina Systems.

San Toribio 6.
28031 Madrid.
ail: com@dadvina.com

Para más información o realizar pedidos contacte con el telefono 91 380 32 46
o en nuestra Web: www.dadvina.com.

* IVA no incluido.

```
BYTE lfPitchAndFamily;
BYTE lf [LF_FACESIZE];
} LOGFONTA;
```

TERMINOLOGÍA USADA

A continuación es el momento de pasar a detallar algunos términos de estos parámetros modificables de las dos estructuras que quizás no queden muy claros para el lector, pero que tras esta explicación esperamos que quede todo perfectamente claro:

- *Ascendente*: El ascendente es la distancia que hay entre las filas de caracteres contando desde la línea de base de la celda de carácter hasta la parte superior de la misma.
- *Línea de base:* La base en que reposan los caracteres, la cual suele estar un poco por debajo de la mitad vertical de la celda de carácter, ya que debe quedar espacio por debajo para poder representar letras con "salientes inferiores" como la "p" "j", etc.
- *Descendente:* Es la distancia que hay desde la línea base hasta la parte superior de la celda de carácter.
- *Altura:* Es la encargada de determinar el tamaño vertical total que posee una celda de carácter de la fuente.
- *Origen:* Corresponde a la esquina superior de la celda de carácter.
- *Ancho:* Tamaño X de la celda de carácter.
- *tmInternalLeading:* Define la cantidad de espacio a insertar dentro

Listado 3. Ejemplo de programación de diversos fuentes.

```
#include <windows.h>
#include <string.h>
LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);

char szProgName[] = "ProgName";

int WINAPI WinMain(HINSTANCE hInst, HINSTANCE hPreInst, LPSTR lpszCmdli-
ne, int nCmdShow)
 {
  HWND hWnd;
  MSG lp;sg;
  WNDCLASS WcApp;

 wcApp.lpszClassName  = szProgName;
 wcApp.hInstance      = hInst;
 wcApp.lpfnWndProc    = WndProc;
 wcApp.hCursor        = LoadCursor(NULL,IDC_ARROW);
 wcApp.hIcon          = NULL;
 wcApp.lpszMenuName = NULL;
 wcApp.hbrBackground = GetStockObject(LTGRAY_BRUSH);
 wcApp.style          = CS_HREDRAW | CS_VREDRAW;
 wcApp.cbClsExtra     = 0;
 wcApp.cbWndExtra     = 0;
  If(!RegisterClass (&wcApp))
    Return 0;

hWnd= CreateWindow(szProgName, "Ejemplo 2. Uso de fuentes en Windows",
WS_OVERLAPPEDWINDOW, CW_USEDEFAULT, CW_USEDEFAULT, CW_USE-
DEFAULT, CW_USEDEFAULT, (HWND) NULL,  (HMENU) NULL, (HANDLE)
hInst, (LPSTR) NULL);

 ShowWindow(hWnd, nCmDShow);
 UpdateWindow(hWnd);
 While (GetMessage(&lpMsg, NULL, 0, 0))
     {
       TranslateMessage(&lpMsg);
       DispatchMessage(&lpMsg);
     }
  return(lpMsg.wParam);
}

LRESULT CALLBACK WndProc(HWND hWnd, UINT messg, WPARAM wParam,
LPARAM lParam)
 {
 HDC hdc;
 PAINTSTRUCT ps;
 static HFONT h0Fony, hNFont;
 static LOGFONT lf;
 char szH1String[]  = "CreatFont( ) cadena horizontal";
 char szV1String[]  = "CreatFontIndirect( )";

 switch (messg)
  {
   case WM_PAINT:
      hdc = beginPaint(hWnd, &ps);
```

Listado 3. (Continuación).

```
    hNFont = CreatFont(25,0,0,0, FM_NORMAL, FALSE, FALSE, FALSE,
ANSI_CHARSET, OUT_DEFAULT_PRECIS, CLIP_DEFAULT_PRECIS,
DEFAULT_QUALITY, 34, "Arial");
    hOFont= SelectObject(hdc, hnFont);
    TextOut(hdc, 300, 100, szH1String, strlen(szH1String));

    lf.lfHeight=25;
    lf.lfWeight=FW_NORMAL;
    lf.lfEscapement= 900;
    lf.lfOrientation=0;
    lf.lfCharSet= ANSI_CHARSET;
    lf.lfPitchAndFamily=49;
    hNFont = CreateFontIndirect(&lf);
    SelectObject(hdc, hNFont);
    TextOut(hdc,300,200,szV1String, strlen(szV1String));

    SelectObject(hdc, HOFont);
    DeleteObject(HNFont);

    ValidateRect(hWnd, NULL);
    EndPaint(hWnd, &ps);
    break;

  case WM_DESTROY:
    DeleteObject(hNFont);
    PostQuitmessage(0);
    break;

  default:
    return(DefWindowProc(hWnd, messg, wParam, lParam));
  }

return(0);
}
```

de la celda de carácter para caracteres que tengan acento, diéresis y símbolos extranjeros.

- *lfPitchAndFamily:* Se utiliza para conseguir definir el número de caracteres de una fuente en particular.
- *tmDigitizedAspecX / tmDigitizedAspecY:* Se usa para crear fuentes tipo raster.
- *tmOverHang:* Acudimos a su uso para sintetizar fuentes, lo cual se requiere cuando una fuente no está disponible en el dispositivo.

FUNCIONES PARA CREAR FUENTES

Casi todos los programadores hacen uso de dos funciones de ámbito muy similar que se para crear sus fuentes que son: *CreateFont( )* y *CreateFontIndirect( ).*

Ambas funciones tienen un funcionamiento bastante similar, pero marcan sus diferencias en la forma de pasar los atributos de definición del tipo de fuente.

CreateFont( ): Esta función pasa los atributos del fuente a crear de forma directa, o sea, indicados dentro del paréntesis. Su sintaxis es la siguiente:

```
HFONT CreateFont( Height, Width, Escapement, Orientation, Weight, Italic, Underline, StrikeOut, Charset, OutputPrecision, ClipPrecision, Quality, PitchandFamily, Facename)
```

Esta función es muy simple de usar, aunque tiene el inconveniente de que cada vez que se quiere definir una fuente tienen que pasarse todos los parámetros a "mano". Para solucionar este problema disponemos de la otra función mencionada: CreateFontIndirect( ).

CreateFontIndirect( ): Esta función, como se ha dicho, no necesita pasar los parámetros manualmente, sino que hace uso de la estructura LOGFONTA, por lo que solamente necesitaremos definir los parámetros una vez.

PROGRAMA DE USO DE FUENTES

A lo largo del Listado 3 se ha incluido un ejemplo bastante sencillo, de uso de fuentes diversas características que una vez que se ejecuta dibuja una cadena de texto horizontal y otra vertical.

Para hacer uso de todo lo explicado con anterioridad, la fuente creada para escribir la primera cadena, se ha creado usando CreateFont, mientras que la segunda se ha realizado con *CreateFontIndirect( ).*

Nada más hechar un vistazo al código, puede comprobarse como no requiere ni ser explicado, debido a su gran sencillez.

Programación del API de Windows (III)

Jordi Agost (agost@eup.udl.es)

En este artículo comenzamos una serie de ejemplos prácticos sobre el mundo de la API, después de haber visto su funcionamiento teórico tal y como explicamos en entregas anteriores. Este es el momento de pasar a la acción y practicar con nuestras propias aplicaciones.

La mayor parte de las aplicaciones realizadas a partir de Visual Basic, por no decir todas, contienen menús para la introducción de diversas órdenes por parte del usuario. A continuación vamos a ver cómo podemos usar la potencia de la API para lograr un grado superior de personificación de los mismos, y así mejorar el aspecto funcional y visual de nuestro programa.

En el entorno Windows tenemos dos tipos de menús: el menú de nivel superior y el emergente. El menú llamado de nivel superior es el que vemos siempre en una ventana, y a partir del cual se despliegan los otros, que cuales desaparecen al pulsar la tecla *Esc*. Los últimos pueden desplegarse de un menú de nivel superior o bien de otro menú emergente.

Estos conceptos son absolutamente válidos sea cual sea la versión de Windows utilizada, aunque los programadores de Visual Basic que programan en 16 bits o hace poco tiempo que han cambiado a 32 bits deberán tener en cuenta que Windows 95/98, introdujo algunos cambios en los menús que pueden afectarnos cuando estemos trabajando desde la *API*.

Estos cambios podrían resumirse brevemente en el hecho de haber añadido un contexto de ayuda (para implementar ayuda *online* en los menús), soporte al menú de contexto, la posibilidad de marcar una entrada por defecto, así como la existencia de nuevas funciones *API* para trabajar con los menús. Las funciones a las que se hace referencia son las siguientes: *GetMenuItemInfo*, *InsertMenuItem* y *SetMenuItemInfo*, las cuales permiten la manipulación de un menú a través de estructuras de tipo *MENUITEMINFO*.

En sus manuales técnicos Microsoft aconseja la utilización de dichas estructuras, pero aunque parezca paradójico no es aconsejable su utilización, debido principalmente a que no poseen compatibilidad con las de Windows NT. Siempre que trabajemos con menús desde la *API*, tendremos que intentar evitar mezclar órdenes de Visual Basic y de la *API* que ofrezcan las mismas funcionalidades. Por ejemplo, si desde la *API* modificamos el texto de un menú, dicha operación se hará correctamente pero si posteriormente leemos la propiedad *Caption* desde Visual Basic, esta propiedad contendrá un valor erróneo. En resumen, podemos decir que cuando queramos establecer el valor de las propiedades *Caption*, *Checked*, y *Enabled* de un menú desde la *API*, tendremos que evitar consultar su valor desde el lenguaje de Visual Basic. Para evitarnos problemas, lo mejor será, que en el caso de querer

modificar alguna de las propiedades anteriormente citadas, lo hagamos desde Visual Basic.

Asimismo también deberemos tener mucho cuidado cuando manipulemos la propiedad *Visible*, ya que cuando marcamos esta propiedad en una entrada del menú con el valor **False**, lo que está haciendo Visual Basic internamente es borrar la entrada de su controlador de menú. Pero si este paso u otros que afectan a la creación o destrucción de entradas de un menú lo realizamos desde la *API*, VB, en la mayor parte de casos, pensará que continua teniendo la entrada (cuando en realidad ya no es así porque desde la API ha sido cambiada). Por este motivo, podría ocurrir que si a partir de ese momento cambiamos alguna propiedad de algún menú, este paso no afectará al elemento de menú que deseemos.

Para entender un poco más este proceso podemos fijarnos en la figura 1, donde con un diagrama de módulos observamos el comportamiento anteriormente descrito.

Uno de los aspectos visuales que podemos modificar en los menús, es el espacio que se encuentra en la izquierda del texto de un menú cualquiera, el cual se utiliza para insertar una marca de verificación (✓).

A través de un conjunto de órdenes de la *API* podemos insertar nuestros propios símbolos personalizados, aunque estos símbolos deben tener el mismo tamaño que la marca normal de Windows. Si desconocemos el tamaño de las marcas estándar de Windows, siempre podemos obtener dichas dimensiones a través de la función de la API *GetMenuCheckMarkDimensions*.

Asimismo lo que podemos conseguir es cambiar un elemento de un menú por el mapa de bits que deseemos, con lo que se redimensionará automáticamente para dar cabida a nuestro propio gráfico.

FUNCIONES CON EL RATÓN

Damos a trabajar con instrucciones de la API que tienen como fin el poder modificar el cursor. Observemos en primer lugar la sentencia *ShowCursor*. Esta función nos da la posibilidad de que podamos elegir entre ver o no ver el cursor en la pantalla. Si deseamos tal efecto, en la sección de declaraciones añadiremos:

```
Private Declare Function ShowCursor& Lib
    "user32" (ByVal bShow As Long)
```

Y en el procedimiento, si lo que deseamos es ocultar el cursor deberemos añadir:

```
re& = ShowCursor&(0)
```

En caso contrario tendremos que añadir:

```
re& = ShowCursor&(1)
```

Para fijar la posición del cursor utilizaremos las funciones *SetCursosPos* y *GetCursorPos*. La primera moverá el cursor hacia una posición determinada y la segunda nos informará de la posición que ocupa el cursor en ese momento. Por ejemplo, en el siguiente fragmento de código se aprecia una rutina que sirve para colocar el cursor en una posición concreta. Dentro de la sección de declaraciones tenemos:

```
Private Declare Function ClientToScreen& Lib
    "user32" (ByVal hwnd As Long, lpPoint
    As POINTAPI)
Private Declare Function SetCursorPos& Lib
    "user32" (ByVal x As Long, ByVal Y As
    Long)

Private Type POINTAPI
  x As Long
  Y As Long
End Type
```

Y en el procedimiento:

```
Public Sub cmdSetPos_Click()
Dim Poi As POINTAPI
```

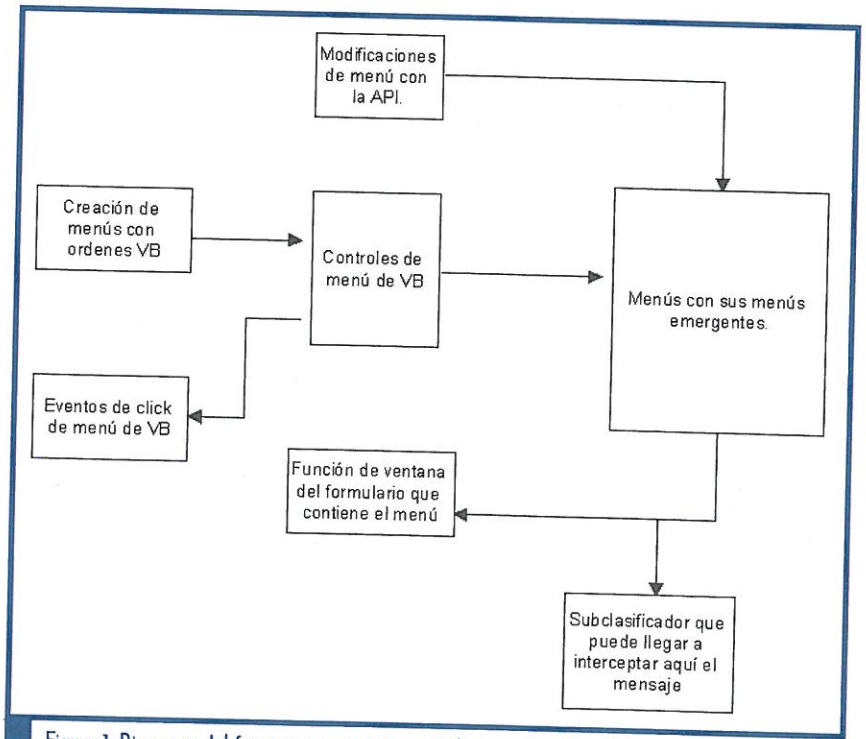


Figura 1. Diagrama del funcionamiento interno de los menús en Visual Basic.

```
Dim re& As Long

  Poi.x = 12
  Poi.Y = 12
  re& = ClientToScreen&(pctCursor.hwnd,
      Poi)
  re& = SetCursorPos&(Poi.x, Poi.Y)
End Sub
```

Otra de las funciones que podemos realizar con el cursor a través de la *API* de Windows *ShowCursor*. Esta función hace que podamos elegir entre ver o no ver el cursor en la pantalla. Utilizaremos básicamente dos funciones para lograr tal cometido: por una parte la función *ClipCursor*, la cual especificará el rectángulo donde quedará circunscrito el cursor y por otra parte *GetClipCursor*, que a su vez recuperará el área actual de circunscripción del cursor (suponiendo que previamente existía un área de este tipo). Vamos a ver un ejemplo del código que realiza dicha función.

En la parte de declaraciones:

```
Declare Function ClientToScreen Lib "user32"
      Alias "ClientToScreen" (ByVal hwnd As
      Long, lpPoint As POINTAPI) As Long
Declare Function ClipCursor Lib "user32" Alias
      "ClipCursor" (lpRect As Any) As Long

Private Type POINTAPI
   x As Long
   Y As Long
End Type

Private Type POINTAPI
   left As Long
   top As Long
   right As Long
   bottom As Long
End Type
```

Y en el procedimiento:

```
Dim Rec As RECT
Dim Poi As POINTAPI
Dim dl&

Poi.x = 0
Poi.y = 0
dl& = ClientToScreen&(pctCursor.hwnd, Poi)
Rec.Top = Poi.Y
Rec.Left = Poi.x
Rec.Right = Rec.Left + pctCursor.ScaleWidth
Rec.Bottom = Rec.Top + pctCursor.Scale-
      Height
dl& = ClipCursor&(Rec)
```

Vamos a examinar el código atentamente. En primer lugar declaramos una variable de tipo *RECT* que servirá para especificar el rectángulo, que mediante coordenadas de pantalla en pixels describirá el área donde se encuentra ubicado el cursor. Declaramos luego una variable de tipo *POINTAPI* que servirá para almacenar el punto donde se encuentra el cursor. Y finalmente declaramos una variable de tipo *long* (dl&) para almacenar el retorno de la función *ClipCursor*. Inicializamos la variable *POINTAPI* a **0**, y luego pasamos las coordenadas de cliente (es decir las internas del ratón que no coinciden con los puntos que vemos por la pantalla) a coordenadas de la pantalla (pixels). A continuación damos valor a la variable **Rec** y finalmente llamamos a la función *ClipCursor* para circunscribir el área.

Si lo que deseamos es inicializar el área de acotamiento, es decir, quitar cualquier área que previamente hubiésemos establecido, deberemos tener en cuenta que la función *ClipCursor* esperará un parámetro nulo para así inicializar el estado de acotamiento. Y debido a que dicha función espera un parámetro *RECT*, para lograr pasar un valor nulo deberemos utilizar un alias de la función que utilice un parámetro *long*, de la forma siguiente:

```
Dl& = ClipCursorBynum&(0)
```

Logrando así inicializar la región de acotamiento.

Para finalizar este apartado nombramos, sin especificar demasiado, algunas de las funciones que pueden resultar de utilidad en alguna que otra ocasión en el tratamiento del cursor. Tenemos por una parte la función *SetDoubleClickTime* que permite fijar el intervalo del doble clic. *SwapMouseButton* intercambiará los botones derecho e izquierdo y finalmente *mouse_event* simulará cualquier evento de ratón.

CAPTURA DE PANTALLA

Cuando un usuario pulsa una tecla se envía a Windows lo que se denomina un **código de exploración.** Este código es un número mediante el cual Windows sabe que una tecla ha sido pulsada, y además dicho código dependerá del teclado en el cual haya sido pulsada la tecla. Es decir, dos teclados distintos, pueden producir diferentes códigos de exploración para una misma tecla (por ejemplo "A").

A fin de hacer una capa lógica de abstracción del teclado, Windows ha creado lo que se conoce con el nombre de **teclas virtuales.** Cuando al sistema le llega un código de exploración, éste convierte la información de un código de exploración en una tecla virtual. Dicha tecla virtual será la que Windows utilizará en su mensaje interno de sistema *WM_KEYDOWN*, y asimismo será la misma que el entorno de Windows trasladará al evento *KeyDown* de un determinado control de Visual Basic. Posteriormente dicha tecla virtual será convertida a un carácter *ASCII* para usarlo en el mensaje interno de Windows *WM_CHAR* y en el evento *KeyPressed* de Visual Basic. En la figura 2 podemos observar el proceso tal y como se ha explicado, y en la figura 3 podemos ver como sería el mismo proceso con un ejemplo real de la pulsación de la tecla "A" de un teclado cualquiera.

Según la documentación de las API de Win32, para capturar una pantalla simplemente debemos utilizar la

tecla virtual *VK_SNAPSHOT* como primer parámetro de la función *Keybd_event*, y en el segundo parámetro poner un **0** si queremos capturar toda la pantalla o bien un **1** si lo que queremos es capturar la ventana activa. A pesar de la información que proporciona la documentación de las *APIs*, dicho procedimiento no funciona en Windows NT. Por este motivo, la mejor solución consistirá en simular el proceso tal y como lo haría un usuario normal, es decir, tenemos que pulsar simultáneamente la tecla *Alt* y la tecla de imprimir pantalla. Continuando con los "fallos técnicos" de la documentación nos encontramos que indica que Windows 95 captura toda la pantalla si en la función ponemos como parámetro 0, cuando en realidad es lo contrario.

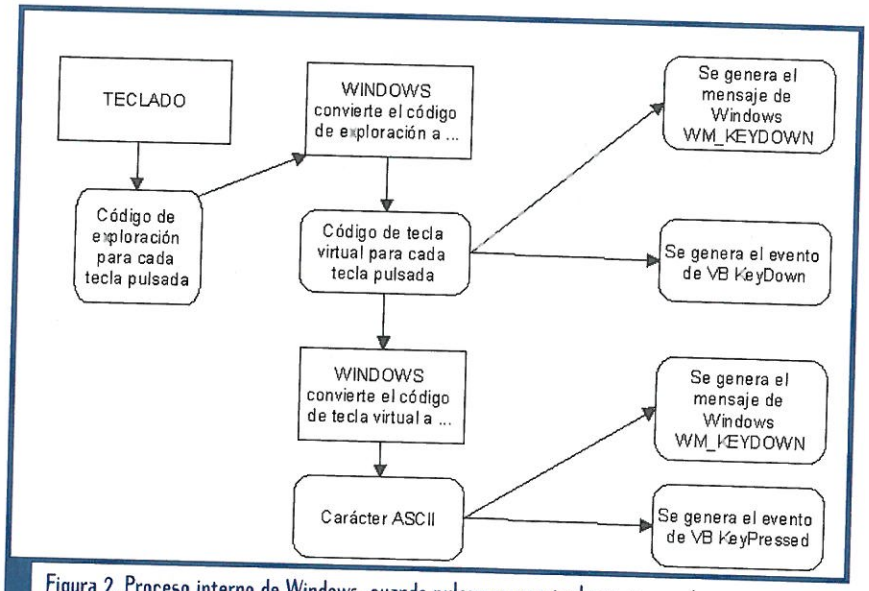


Figura 2. Proceso interno de Windows, cuando pulsamos una tecla en un menú.

De lo explicado anteriormente deducimos que para realizar una rutina de tipo "universal", es decir, que pueda ejecutarse en cualquier entorno Win32 bits, necesitaremos una rutina que indique el sistema en el cual estamos ejecutando la aplicación (Windows 95/98 o Windows NT), ya que en función del mismo, tendremos que actuar de una orma o de otra. Para esto bastará con stas pocas líneas de código:

```
)im dl
)im inf As OSVERSIONINFO
if.dwOSVersionInfoSize = Len(inf)
l& = GetVersionEx(inf)
 inf.dwPlatformId =
    VER_PLATFORM_WIN32_WINDOWS
    Then
quí ponemos a true la variable, que indica que
    estamos en W95
nd If
```

En ellas definimos primero una riable para retorno de la función de la *LL* (dl&), y otra variable del tipo *SVERSIONINFO*. A continuación re-namos parte de la información de la tructura (la referente al tamaño) y mamos a la función *GetVersiónEx* que volverá en la estructura toda la información referente a la versión del sistea operativo que estamos utilizando.

Lo primero que necesitaremos en la rutina son las declaraciones:

```
Dim Alt%, vari%
```

Y el código de exploración después de la tecla *Alt* para utilizarlo posteriormente si tenemos que simular su pulsación (para capturra toda la pantalla):

```
Alt% = MapVirtualKey(VK_MENU,0)
```

Luego si es pantalla completa deberemos que simular la pulsación de la tecla *Alt*. Utilizamos la función *Keybd_event* para tal cometido:

```
If ..... Then 'aquí añadimos la condición de si es
    pantalla activa o completa.
    Keybd_event VK_MENU, Alt, 0, 0
End If
```

Una vez realizado dicho paso, comprobamos si estamos en Windows 95 o no (podemos utilizar la rutina dada anteriormente) y en caso afirmativo ponemos una variable a 1.

```
If ... Then 'las condiciones son pantalla parcial
    = True y Windows95 = True
vari% = 1
End If
```

Ahora simulamos la pulsación de la tecla de captura de pantalla. Nótese que el segundo parámetro es el que hemos modificado en el párrafo anterior y varia según nos encontremos en W95/98 o en Windows NT.

```
Keybd_event VK_SNAPSHOT, vari, 0, 0
```

Pero aún tenemos la tecla *Alt* "pulsada" por lo que deberemos dejar de hacerlo. Para ello deberemos hacer uso de la misma función *Keybd_event*, pero en el tercer parámetro le decimos que deje de pulsar la tecla. De esta forma se finalizar el proceso tal y como vemos a continuación:

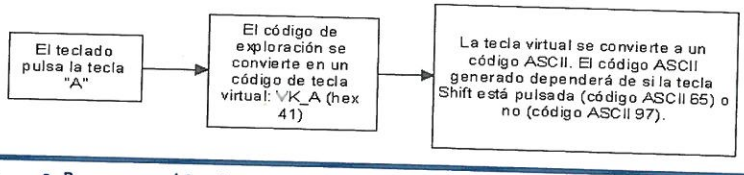


Figura 3. Proceso simplificado cuando pulsamos la tecla "A" en un teclado.

```
If ..... Then 'aquí añadimos la condición de si es
        pantalla activa o completa.
        Keybd_event VK_MENU, KEYE-
        VENTF_KEYUP, 0, 0
End If
```

VARIABLES DE ENTORNO

Para un programador puede ser de cierto interés conocer el contenido de las variables de entorno en un momento dado, a pesar de que dentro de la arquitectura Windows de 32 bits, temas tales como los ficheros de inicialización o las variables de entorno van siendo desplazados poco a poco por el uso del registro. Pero aún así los sistemas Windows de 32 bits (léase Windows 95, Windows 98 y Windows NT) todavía hacen uso de las variables de entorno para lograr controlar el funcionamiento de algún aspecto del sistema.

Podemos acceder desde VB a las variables de sistema a través de la función *GetEnvironmentVariable*, la cual devolverá una cadena con todas las variables de entorno. El análisis posterior será misión del programador, que tendrá que trabajar con el contenido de la cadena con el fin de ir extrayendo una a una todas las variables de entorno, ya que dentro de la misma se aparecen todas consecutivas). Un ejemplo del uso de dicha función sería el siguiente:

```
Public Function GetStringValue$(ByVal envstr$)
Dim re&
Dim buf$

   buf$ = String$(2048, 0)
   re& = GetEnvironmentVariable(envstr$,
      buf$, 2047)
   If re& > 0 Then 'análisis por parte del pro-
      gramador
End Function
```

En esta función hemos declaramos dos variables, la primera de tipo *long* (re&) que servirá para almacenar el valor que devuelve la función y otra de tipo *string*. Si este valor es mayor que **0** querrá decir que la función se ha ejecutado correctamente y ya podemos leer el contenido de la variable (sentencia *If*). La otra variable que declaramos, la de tipo *string*, servirá para almacenar la cadena que devolverá la función *GetEnvironmentVariable* en el caso de que se haya ejecutado correctamente.

EJECUCIÓN DE UN PROGRAMA

Muchas veces nuestros programas deben lanzar tareas (a veces incluso módulos que funcionan en ventanas DOS) y proseguir cuando dichas tareas finalicen. Anteriormente en los sistemas de 16 bits se usaban funciones tales como *FindWindow*, para encontrar el nombre de la ventana o a veces *GetNumTasks* para saber el número de tares que se estaban ejecutando en un momento dado. Con la *API* de 32 bits el sistema ha cambiado de forma radical. Vamos a ver el procedimiento recomendado por Microsoft para la ejecución de tareas externas a nuestro programa. Para ejecutar un programa o tarea desde la *API32*, en primer lugar necesitamos dos estructuras: *PROCESS_INFORMATION y STARTUPINFO*.

Veamos a continuación detenidamente la estructura *PROCESS_INFORMATION:*

```
Private Type PROCESS_INFORMATION
  HProcess As Long
  HThread As Long
  DwProcessId As Long
  DwThreadId As Long
End Type
```

La utilizaremos en la función *CreateProcess*, que es la que lanzará el proceso y servirá para que dicha función pueda devolver información sobre el proceso que acabamos de crear. En dicha estructura se almacenará el manipulador que identifica el nuevo proceso, así como el manipulador para la trama principal.

La estructura *STARTUPINFO* se utiliza para que la función *CreateProcess* pueda controlar todos los aspectos relativos a la creación del proceso. Vamos a examinar ahora el procedimiento con detalle:

```
Public Sub ExecCmd(cmdline$, tipus%)

Dim proc As PROCESS_INFORMATION
Dim start As STARTUPINFO
Dim ret As Long

 start.cb = Len(start)
 ret = CreateProcessA(0&, cmdline$, 0&, 0&,
      1&, NORMAL_PRIORITY_CLASS, 0&,
      0&, start, proc)
 ret = WaitForSingleObject(proc.hProcess,
      INFINITE)
 ret = CloseHandle(proc.hProcess)

End Sub
```

En primer lugar se han declarado las dos estructuras necesarias y una variable de tipo *long* que servirá para guardar los valores de retorno de las funciones de la *API*. Con la sentencia:

```
start.cb = Len(start)
```

Inicializamos la estructura *STARTUPINFO*, lanzamos el proceso y a continuación la función *WaitforSingleObject* suspenderá la trama principal del programa hasta que el proceso lanzado por la función *CreateProcess* sea cerrado. Si nos detuviésemos aquí la aplicación quedaría paralizada por lo que es absolutamente necesario que cerremos el manipulador del proceso para así permitir que Windows borre el objeto del proceso que se ha creado.

Unix vs Windows NT

Fco. Javier Toledo (Jtoledo@sei.es)

¿Debería optar por una solución basada en un sistema Unix o en Windows NT? ¿Cuál es mejor sistema operativo?. Desde luego la respuesta no es sencilla, pero en este artículo trataremos de obtenerla basándonos en las comparaciones, que aún siendo odiosas, suelen convertirse en un buen punto de referencia.

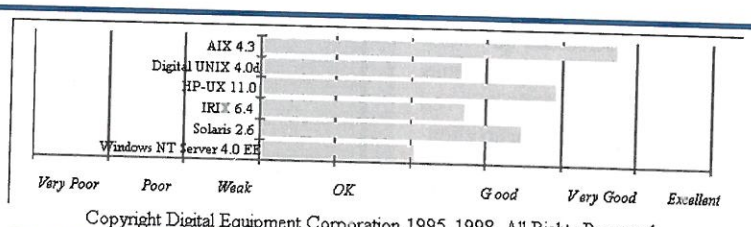
INTRODUCCIÓN

Todos los administradores de sistemas se han planteado alguna vez si optar por Unix o por Windows NT, nos referimos a una realidad que lleva mucho tiempo en debate. La solución se encuentra en el criterio de cada uno y en las necesidades que se planteen en cada caso. Un administrador de Windows NT seguramente argumentará que Unix es viejo, con un sistema de órdenes críptico y caduco. Sin embargo un administrador Unix nos dirá que NT no es fiable y que no se adapta bien a las necesidades actuales.

Los sistemas operativos ideados para dar servicio a los PC's y a los pequeños ordenadores se han abierto un hueco en el mundo empresarial e industrial. Windows NT 4.0. es una buena muestra de la evolución que ha sufrido la arquitectura PC, aunque buena parte ha tenido que ver con la agresiva política de Microsoft. Pero el triunfo de NT, aunque ayudado por ella, se debe principalmente a sus prestaciones en conectividad Internet, sistema de 32 bits, seguridad, escalabilidad, consistencia, multitarea, portabilidad, compatibilidad, tolerancia frente a fallos, etc.

WINDOWS NT

El primer miembro perteneciente a la familia Windows distribuido a gran escala en el mercado fue la versión Windows 3.1, el siguiente paso abordó la falta de interconectividad entre los distintos usuarios, de tal forma que Microsoft desarrolló Windows para trabajo en grupo o Windows 3.11 que permitía compartir directorios, impresoras y otros dispositivos en una red local. El siguiente paso de Microsoft fue construir un sistema operativo que ofreciera mayores prestaciones en el ámbito de la conectividad, dotado de una mayor robustez. Windows NT Server es un servidor que incorpora aplicaciones de gestión de red. En el último trimestre de 1996 se sumó a la familia Windows la última versión de este servidor (Windows NT 4.0) que incorpora la interfaz de usuario propia de Windows 95 y servicios de conectividad con otras redes y con Internet.



Copyright Digital Equipment Corporation 1995-1998. All Rights Reserved.

Figura 1. Comparativa de la administración de sistema. Extracto de Digital Equipment Corporation's. Del artículo AIX 4.3 Leaps To 64-Bits In Dead Heat With Digital UNIX 4.0.

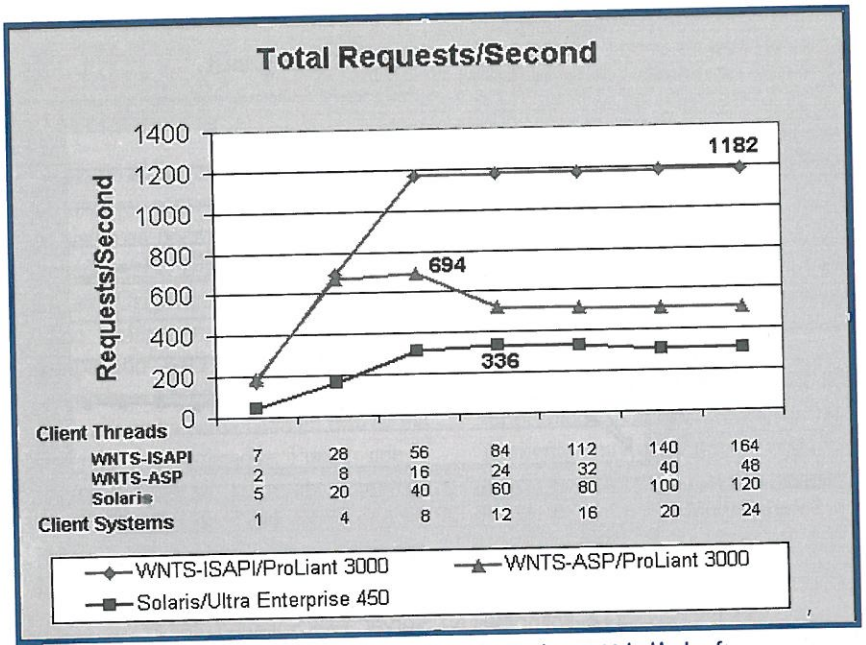


Figura 6. Comparación web de peticiones segundo. Copyright © 1998 by Mindcraft.

- Requiere que los usuarios se identifiquen introduciendo un único y exclusivo identificador de acceso y una contraseña antes de que le sea permitido el acceso al sistema.

- El propietario de un recurso puede determinar quién puede acceder al recurso y lo que puede hacer con él. Dicho dueño consigue esta característica garantizando derechos de acceso a un usuario o a un grupo de usuarios.

- La auditoría proporciona la posibilidad de detectar y registrar eventos importantes relacionados con la seguridad al crear, acceder o eliminar recursos del sistema. Utiliza identificadores de acceso para registrar de identidad del usuario que llevó a cabo la acción.

- La protección de memoria evita que cualquiera lea información escrita por otra persona, incluso después de que una estructura de datos haya sido devuelta al sistema operativo. La memoria es reinicializada antes de volver a ser utilizada.

El sistema operativo Unix está equipado con un shell más versátil que NT

Los *bugs* de NT en este sentido han sido mas conocidos y han dado algún que otro dolor de cabeza a los administradores. Pongamos como ejemplo el ataque a los servidores web mediante el método *denial of service*, también conocida como Bonk, Boink y New Tear. El método se valía de un fallo de Windows NT, que deja "colgados" a los servidores al obligarles a autobombardearse con información. El problema se solucionó con un parche que inmediatamente sacó Microsoft.

SISTEMA DE ARCHIVOS

NTFS es el sistema de archivos de NT, aunque también puede trabajar con el sistema tradicional de FAT utilizado de forma habitual, aunque el primero admite un mayor grado en el control de acceso del que es posible bajo Unix. En NTFS, el usuario que creó el archivo o el directorio es el único que tiene permisos para acceder, modificar o cambiar los permisos de los mismos. Un archivo posee además de los permisos una entrada de lista de control de acceso (ACL: Access Control List) que define los accesos de usuarios o grupos a los archivos o directorios.

PERMISOS DE ARCHIVO DE WINDOWS NT

El propietario puede conceder los siguientes permisos a un usuario, a sus archivos o directorios:

En cuestión de seguridad ambos sistemas obtienen una alta calificación

Para los archivos y directorios:

- Sin acceso alguno.
- Lectura. El usuario puede leer y ejecutar, pero no se le permite la modificación.
- Cambio. El usuario puede ver, leer, ejecutar, modificar, cambiar atributos y eliminar archivos o directorios.
- Control total. Leer, ejecutar, modificar, eliminar archivos y tomar el control sobre ellos.
- Acceso especial. Permite conceder el permiso P a la creación per-

Tabla 1. Características de Windows Nt y Unix.

	WINDOWS NT SERVER 4.0	UNIX
Estabilidad	Buena, comparado con los demás sistemas operativos de PC es bastante fiable, pero no alcanza los niveles ofrecidos por las distintas versiones y Unix.	Muy buena, es uno de los puntos fuertes de este sistema operativo. Podemos considerarlo como de estabilidad probada.
Facilidad de uso	Muy buena, gracias a la filosofía de Microsoft que apoya las buenas interfaces mediante las que se facilita la vida al usuario.	Requiere de sus usuarios un mayor grado de especialización, puesto que utilizar una shell de Unix no es una tarea trivial.
Fabricantes	Microsoft es el único propietario, por lo cual la evolución de este sistema operativo está sometida a las decisiones de una única compañía.	Múltiples fabricantes. Esto implica que los avances se realizan en más de un frente.
Administración	Buena, la administración de toda la red puede realizarse desde un único equipo y está apoyada por la facilidad que aportan las interfaces de Windows, aunque aún es obligatorio reiniciar el servidor ante las nuevas instalaciones o cambios en la configuración.	Muy buena, no requiere reiniciar el servidor ante los cambios de configuración o nuevas instalaciones. Destacan las aplicaciones que facilitan la vida al administrador tales como cron y el telnet.
Telnet	Sólo la parte cliente.	Sí, tanto la parte del cliente como la del servidor.
Automatización	Mala, pues se debe recurrir a aplicaciones externas para poder desarrollar guiones complicados y programar los eventos temporales.	Muy buena, las aplicaciones necesarias para la automatización ya vienen incorporadas.
Variantes entre versiones	Pocas, al ser un único fabricante se mantiene la coherencia.	Bastantes, existen muchas diferencias sutiles entre las distintas versiones de los distintos fabricantes.
Seguridad	Buena, cumple con el estándar de seguridad C2.	Buena, la mayoría de las distintas versiones comerciales de Unix cumplen con el estándar de seguridad C2.
Limitación del espacio de disco de usuario.	Windows NT Server no dispone actualmente de esta posibilidad (no viene por defecto en el sistema operativo) pero ésta es otra de las muchas mejoras que se pretenden abordar con NT 5.0.	Sí.
Rendimiento	Bueno.	Muy bueno, superando incluso al de NT. Una ventaja más, el mayor número de procesadores que permiten algunas versiones de Unix, así como que se trata de un sistema operativo que trabaja con 64 bits.

Tabla 1. Características de Windows Nt y Unix.

	WINDOWS NT SERVER 4.0	UNIX
Soporte de multiprocesadores simétricos	Sí.	Sí.
Concurrencia	Sí.	Sí.
Apoyo al desarrollo	Muy bueno, el inmenso número de aplicaciones existentes para desarrolladores en este sistema es apoyado por la calidad de las mismas y la velocidad con que Microsoft actualiza sus diferentes herramientas.	Bueno.
Precio	Económico, el precio de este sistema operativo es mucho más económico que el de las versiones Unix, pero debemos tener en cuenta que si no añadimos software adicional (por ejemplo BackOffice) únicamente vamos a poder explotarlo como servidor de ficheros y/o impresoras.	Costoso, pero sin duda alguna rentable. Incluye por defecto un software más completo que permite explotarlo directamente como servidor de impresoras, archivos, correo. Además permite trabajar frente a un cliente mediante telnet, como si estuviésemos en consola del servidor.
Rsh	SI. RSHSWC es el equivalente al rsh de Unix, aunque no distribuido como parte de NT Server 4.0 se distribuye con el Kit de recursos de Windows NT.	Sí.
Control de acceso en el sistema de archivos	Muy bueno, admite mayor grado de configuración en el control de acceso.	Bueno.
Fiabilidad	Los procesos se ejecutan en espacios de memoria distintos y el fallo de uno de ellos no provoca la caída del otro.	Muy buena, podemos considerarlo como de fiabilidad probada.

sonal de una entrada de control de acceso para una ACL.

Exclusivos para directorios:

- Listado. El usuario puede ver el contenido del directorio pero no puede acceder a los contenidos.
- Agregar. El usuario puede añadir un archivo al directorio.

Permisos de grupo:

Existen dos tipos de grupos en Windows NT: los locales y los globales. Los grupos locales son aquellos que quedn definidos por el usuario y se usan para asignar permisos a los recursos locales. Un grupo local puede contener usuarios y grupos globales. Para facilitar la gestión de los grupos existen grupos predeterminados con una serie de derechos ya preestablecidos.

Para cada archivo y directorio se almacena el identificativo del usuario y del grupo

PERMISOS DE UNIX

En Unix existen tres categorías de permisos: usuarios, grupo y otros. A cada categoría se le pueden asignan los privilegios de lectura, escritura y ejecución. Para cada archivo y directorio se almacena el identificativo del usuario y del grupo (UID y GID), de tal forma que la combinación de los permisos y privilegios da como resultado el acceso de los usuarios al sistema de archivos.

RENDIMIENTO

El rendimiento (la velocidad de procesamiento) depende en mayor grado del hardware que del software. Como ya se comentó anteriormente, los sistemas operativos Unix comerciales han sido históricamente empleados en grandes sistemas, mientras que Windows NT está siendo explotado principalmente en las plataformas x86.

El rendimiento depende en mayor grado del hardware que del software

El rendimiento de estos dos sistemas operativos de servidores está ajustado para dar buenos tiempos como servidores de archivos, impresión y aplicaciones. La versión comercial de Windows NT Server admite hasta cuatro procesadores en un entorno de multiproceso simétrico. Pero las implementaciones de equipos originales (OEM) de Windows NT Server admiten hasta 32 procesadores en este entorno. Mientras que el número de procesadores en los diferentes Unix varía superando en algunos casos esta cifra.

DESARROLLOS

En este punto NT se encuentra en cabeza con respecto a Unix, ya que el mercado de aplicaciones para Windows lleva ya mucho tiempo disparado.

Las apuesta más fuertes en el ámbito de Internet se decantan en favor de Unix

La gran variedad de lenguajes disponibles para desarrollar aplicaciones que corran en sistemas Windows y las herramientas añadidas que contienen, hacen que los desarrolladores se inclinen hacia este lado al elegir un destino para sus aplicaciones. Dado que, tanto Windows 95 como NT son dos sistemas operativos de 32 bits, las aplicaciones pueden correr en cada uno de ellos siendo necesario un mínimo esfuerzo.

SOLUCIONES DE LAS GRANDES EMPRESAS

En la actualidad las grandes empresas realizan sus apuestas en diversos frentes, pero parece ser que las apuestas más fuertes en el ámbito de Internet se decantan en favor de Unix. Www.amazon.com es una de las direcciones de Internet que seguramente ya hemos visitado, y sobre ella podemos comentar que ha confiado sus necesidades en un sistema DIGITAL Unix AlphaServer 2000.

Al parecer en el ámbito de Internet, cuando se requiere alta potencia y rendimiento, Unix aún tiene ventaja sobre NT. Prueba de ello es el caso de Hotmail que ofrecen un servicio gratuito de email.

En Diciembre de 1997 Microsoft adquirió esta compañía y como es lógico intentaron migrar los sistemas a NT, pero el número de usuarios de este servicio es inmenso (10 millones de usuarios) y Microsoft tuvo que volver a instalar Solaris.

Ubi Soft
ENTERTAINMENT

¿TE GUSTARÍA TRABAJAR EN UNA COMPAÑÍA JOVEN Y DINÁMICA EN PLENA EXPANSIÓN?

Ubi Soft, multinacional francesa, líder en la edición de vídeojuegos y la aplicación multimedia, implantada en Estados Unidos, Europa y Asia busca ingenieros para estudio y desarrollo para su departamento de I+D ubicado en Sant Cugat del Vallés.

Perfil : Recién Licenciado en informática o en telecomunicaciones, conocedor del lenguaje C. Has realizado recientemente una práctica o un proyecto importante en C o C++. Valoraremos tus conocimientos de desarrollo en Windows y en ensamblador 80x86. Imprescindible buen dominio del Inglés.
Si te interesa la informática lúdica y técnica (inteligencia artificial, red, imagen, arquitectura de programas), ven a incorporarte a un equipo joven y dinámico en una compañía en plena expansión.

Envía tu curriculum y una carta exponiendo tus motivaciones por correo electrónico a ubistudios@ubisoft.es, por fax al número (93)589.56.60 o por correo a UBI SOFT, Plaza de la Unión nº1, 4ª plta – 08190 Sant Cugat del Vallés (Barcelona)

Tabla 2. Especificaciones técnicas de Windows NT 4.

Plataforma cliente/servidor de alto rendimiento	• Sistema operativo multifunción para servicios de archivos, impresión, intercambio de mensajes y aplicaciones, escalable hasta 32 CPUs. • Capacidad multitarea multithread; sistema operativo de proceso multinivel de 32 bits. • Protección de memoria para las aplicaciones. • Independiente del procesador: admite procesadores Intel, MIPS, Alpha AXP y PowerPC. • Admite 4 GB de RAM por sistema; 2 GB de memoria virtual por aplicación; 402 millones de TB de almacenamiento de datos por sistema.
Base de red	• Protocolos y transportes incluidos: TCP/IP, NetBEUI, IPX/SPX, DLC, y AppleTalk®. • Clientes Telnet y FTP, FTP Server, Servicios para el Macintosh. • Administrador de clientes de red para la instalación de software cliente, incluido Windows95. • Servicios integrados de intercambio de mensajes.
Herramientas de gestión	• Administrador de usuarios con seguridad y configuración mediante operación "arrastrar y soltar". • Administrador de disco para configuración gráfica de discos. • Monitor de rendimiento, que analiza el rendimiento del servidor y de la red. • Visualizador de eventos, que registra la actividad del servidor. • Gestión remota desde cualquier PC que ejecute Windows. • Herramientas para monitorizar los sistemas clientes y gestionar las licencias de acceso de cliente.
Servicios NetWare	• Gateway para NetWare: proporciona acceso a servidores NetWare desde sistemas clientes Microsoft. • Herramienta de migración para NetWare: transfiere las cuentas de usuarios y de grupos, los scripts de inicio de sesión y los datos de NetWare a Windows NT Server. • Servicios de archivos e impresión para NetWare: (opc.) permite a los sistemas clientes NetWare acceder a Windows NT Server sin necesidad de modificar el software cliente.
Servicios TCP/IP	• Protocolo de configuración dinámica de host (DHCP): asigna automáticamente las direcciones IP. • Servicios de nombres Internet Windows (WINS): asigna nombres de ordenadores a direcciones IP. • Soporte directo de impresión para impresoras de red basadas en TCP/IP y UNIX.
Servicio de acceso remoto	• Admite acceso remoto con protocolos NetBEUI, IPX/SPX, y TCP/IP, a través de RDSI, X.25 y líneas telefónicas estándard. • Soporte de clientes para UNIX a través de PPP, NetWare, Shiva® LanRovers, Windows 3.x, Windows '95, Windows NT Server, Windows NT Workstation y LAN Manager. • Admite 256 conexiones simultáneas.
Configuración e Instalación.	• Configuración basada en CD-ROM detecta y configura automáticamente miles de dispositivos. • Soporte de scripts para permitir una configuración automatizada y una instalación desatendida. • Permite al usuario revisar y configurar todos los protocolos y servicios de la red.

Tabla 2. Especificaciones técnicas de Windows NT 4. (Continuación).

Seguridad	● Admite seguridad local; exige identificación de usuario y contraseña para acceder al sistema. ● Control por el administrador de bloqueo de cuentas para impedir ilimitados intentos de inicio de sesión. ● Seguridad certificada nivel C2. ● Seguridad NTFS por archivo y por directorio. ● Admite entornos de estaciones de trabajo sin disco mediante la carga remota de programas.
Tolerancia a los fallos	● Sistema de archivos basado en transacciones (NTFS) y duplicación de directorio. ● Admite fuentes de alimentación ininterrumpida. ● Duplicación, réplica de disco y distribución de datos en varios discos con paridad (RAID 5). ● Tolerancia a los fallos/réplica de servidor en tiempo real (opc.).
Copias de seguridad	● Utilidad incorporada de copias de seguridad en cinta. ● Copia de seguridad/restauración del disco del servidor con seguridad. ● Copias de seguridad en línea de los archivos de cuentas de usuarios.
Servicios de directorio	● Un único login en la red para acceder a las aplicaciones de archivos, impresión y servidor. ● Administración centralizada de las cuentas de usuarios y grupos a través de varios servidores. ● Base de datos duplicada de cuentas de usuarios y grupos. ● Independiente del protocolo.

Tabla 3. Información sobre Solaris 2.6.

Lenguajes	C : ● Compilador de C : incluido en todos los Workshop (Excepto en JavaTM Workshop y Pascal Workshop). ● Workshop Pro C : Entorno completo de desarrollo en C. C++ : ● Visual Workshop C++ : Entorno gráfico de desarrollo. FORTRAN : ● Performance Workshop Fortran: Entorno completo de programación en Fortran. ADA : ● Workshop for Ada : Entorno completo de programación en Ada. COBOL : ● Micro Focus Cobol : Entorno completo de desarrollo en Cobol.

Tabla 3. Información sobre Solaris 2.6. (Continuación).

Entorno de Desarrollo	TM/INTERNET : ● JavaTM Workshop : Herramienta de desarrollo para aplicaciones JavaTM. ● Sun Internet WorkShop : Desarrollo de aplicaciones cliente/servidor para Internet/Intranet. ● JavaTM Studio : Herramienta gráfica de desarrollo para aplicaciones JavaTM. ● JavaTM Virtual Machine : El interprete de aplicaciones JavaTM. ● JavaTM JIT Compiler : Compilador "Just In Time" para JavaTM. OBJETOS : ● SolarisTM Neo : Basado en OMG-CORBA, permite llevar la filosofía cliente/servidor a Internet.
Seguridad	● Solstice Security Manager : Productos que garantizan el acceso exclusivo de los usuarios autorizados a los servicios de la red. ● Sun Screen SKIP Encryption Clients : Seguridad de comunicación en redes públicas encriptando todos los paquetes IP. ● Sun Screen EFS : Servicio de encriptación que incluye Firewall y SKIP. ● Sun Screen SPF : Hardware de encriptación. ● Firewall 1 : Seguridad para sus redes, transmisión y usuarios.
Gestión de red	● Solstice Autoclient : Gestión de equipos de sobremesa. ● Solstice Job Scheduler: Gestor de táreas. ● Solstice Site/Domain/Enterprise Manager. ● Gestor de redes que se adapta a las necesidades de su empresa. ● Solstice AdminSuite : Conjunto de herramientas de administración centralizada de SolarisTM. ● Solstice Network Client : Conjunto de herramientas de gestión de PC's.
Gestión Almacenamiento	● Solstice Backup : Backup y Restore de sistemas heterogéneos. ● Solstice DiskSuite : Gestión de almacenamiento (Sun). ● Solstice Symon : Herramienta de monitorización de sistemas. ● Solstice Volume Manager : Gestión de almacenamiento (Veritas) Soporte directo de impresión para impresoras de red basadas en TCP/IP y UNIX.
Conectividad	REDES DE PC'S ● Soltice LM Server : Permite a los clientes LAN Manager acceder a los servicios y aplicaciones SolarisTM. ● Solstice NW Server : Los servidores Sun aparecen como servidores Netware. WAN's ● Solstice OSI : Interacción transparente entre los sistemas OSI. ● Solstice CMIP : Desarrollo de agentes y aplicaciones para redes globales de telecomunicación.

Tabla 3. Información sobre Solaris 2.6. (Continuación).

- Solstice FTAM : Gestión acceso y transferencia de ficheros con otros sistemas FTAM.
- Solstice TMN : Entorno completo de desarrollo para áreas funcionales de la gestión de red en telecomunicaciones.
- Solstice X.25 : Conectividad transparente en redes conmutadas de paquetes de datos.
- Solstice X.400 : Herramientas de creación de redes de intercambio de mensajes.
- Solstice X.500 : Creación de espacios comunes de nombres para servicios de directorio.

OTROS SISTEMAS

- SunLink SNA 3270 : Integración de sistemas SolarisTM/SPARC en redes IBM.
- SunLink C6 3270 : Emulación gráfica del terminal IBM 3x79.
- SunLink SNA 3770/RJE : Transferencia de ficheros con entornos IBM.
- SunLink SNA/LU0 : Desarrollo de aplicaciones para entornos IBM.
- SunLink SNA/SNM : Gestión de redes.
- SunLink Client 3270 : Emulación terminales IBM 3270.
- SunLink SNA/PTP : Runtime para aplicaciones LU 6.2 de IBM.
- SunLink DNI : Integración con entornos DECnet.

COMPARATIVA DE DOS GRANDES COMPAÑÍAS

Tanto Microsoft como Sun Microsystems tienen disponible información de comparativas entre Windows NT Server y Solaris. Evidentemente no podemos esperar encontrarnos valoraciones equivalentes en sus comparaciones, ya que cada uno conoce los puntos fuertes de su sistema operativo y enfocan estas comparativas convenientemente para resaltarlos. Con todo ello, esta información puede resultar muy didáctica a la hora de identificar estos puntos. No nos sorprendamos por tanto por lo que parecen ser contradicciones entre las diversas comparaciones gráficas incluidas en el artículo.

PRUEBAS REALIZADAS POR MICROSOFT

Las pruebas se encuentran disponibles en la dirección de Internet: http://www.microsoft.com/windows/news/august1998/ntperformance.asp?site=nts&custarea=bus

Las principales ventajas de Unix son fiablidad y estabilidad, ya que proporciona facilidades para la automatización

Pruebas de Servidor de Ficheros

"Mindcraft Inc., ha probado el servidor Windows NT 4.0 en un Compaq ProLiant 3000 y encontró que Windows NT Server 4.0 alcanza un mayor rendimiento en la compartición de ficheros comparándolo con Solaris de Sun. Con una configuración equivalente, su rendimiento es más de 4.2 veces el de Solaris 2.6 con TotalNET en un Sun Ultra Enterprise 450. La relación precio/rendimiento en una plataforma Windows NT Server/ProLiant 3000 es 11.2 veces superior a la plataforma Solaris/TotalNET/Ultra Enterprise 450.

Las pruebas probaban el rendimiento en la compartición de ficheros SMB sobre TCP/IP. SMB (Bloque de mensajes del servidor) es el protocolo estándar que utiliza Windows NT para compartir servicios de archivos o de impresión. En el servidor de Sun estaba corriendo TotalNET 5.2 para poder utilizar este protocolo.

Las diferencias entre los dos sistemas tienden a difuminarse ya que NT está avanzando a grandes pasos

Los resultados de esta prueba muestran que NT supera a Solaris en un 4.2 en MB/S y 11.2 en la relación $/MB.

Pruebas de Servidor web

Las pruebas realizadas en servidores web corriendo bajo Windows NT 4.0 y Solaris muestran un rendimiento 4 veces superior para Windows NT 4.0 en contra

Tabla 4. Servidor de Ficheros

SERVIDOR DE FICHEROS	MÁXIMO RENDIMIENTO	PRECIO	PRECIO/RENDIMIENTO
Compaq ProLiant 3000 (2 x 333MHz Pentium II, 512 MB)	8.0 MB/S	$13,900	$1,737.50/MB/S
TotalNET/Solaris 2.6 Sun Ultra Enterprise 450 (2 x 296 MHz Ultra SPARC, 512 MB)	1.9 MB/S	$37,047	$19,498.42/MB/S

Tabla 5. Servidor web.

SERVIDOR WEB	PEAK RPS	MÁXIMO RENDIMIENTO	PRECIO	PRECIO/RENDIMIENTO
Windows NT Server 4.0 (2 x 333MHz Pentium II, 512 MB)	1337 (ISAPI) 694 (ASP)	17.1 MB/S 8.8 MB/S	$12,250	$9.17/RPS (ISAPI) $17.65/RPS (ASP)
Solaris with SWS 1.0 (2 x 296 MHz Ultra SPARC, 512 MB)	336	4.5 MB/S	$31,828	$94.75/RPS

de Solaris. Las pruebas fueron realizadas sobre el mismo hardware anterior y utilizando como servidores web Internet Information Server para Windows NT y Sun WebServer 1.0 para Solaris. La siguiente tabla muestra el rendimiento en megabytes por segundo servidos por HTTP y el precio de cada sistema.

Un entorno mixto puede ser la solución a todos nuestros problemas

Windows NT Server 4.0 en un Compaq ProLiant 3000 suministra un rendimiento superior para webs "pesados" (con muchos accesos y muchas páginas HTML). Esta plataforma genera un rendimiento de páginas dinámicas y estáticas superior a Solaris 2.6 SWS 1.0 en una Sun Ultra Enterprise 450 sirviendo únicamente peticiones de páginas estáticas. Para mantener el equipo ProLiant 3000 trabajando en su máximo rendimiento se necesita un ancho de banda superior a 200 Mbits/segundo de 100Base-TX"

PRUEBAS REALIZADAS POR SUN MYCROSYSTEMS

Las pruebas se encuentran disponibles en la dirección de Internet: www.fisica.uson.com/solaris

CONCLUSIÓN

Las principales ventajas de Unix frente a NT son la fiabilidad y la estabilidad, ya que se trata de un sistema apoyado por múltiples fabricantes que proporciona grandes facilidades para la automatización y desarrollo de guiones. Sus principales desventajas radican en una administración un tanto más complicado que en Windows NT. Los servidores tienden a ser muy caros y existen diversas variedades de Unix con pequeñas diferencias entre ellas. Bajo Windows NT los servidores y estaciones de trabajo tienden a ser más baratos que los de Unix (exceptuando las versiones gratuitas del mismo, ej. Linux), siendo uno de los puntos fuertes de NT la existencia de una gran variedad de aplicaciones de terceras partes.

Podemos decir que hoy en día las diferencias entre los dos sistemas operativos tienden a difuminarse ya que la evolución de NT está avanzado a grandes pasos (todos estamos esperando la llegada de NT 5.0). En cualquier caso, existe cierta ralentización debido a que NT sigue siendo propiedad de Microsoft y se encuentra sujeto a sus planes de

Tabla 6. Pruebas de Sun MycroSystems.

	BENEFICIO	WINDOWS NT	SUN SOLARIS 2.6 PARA INTRANETS	VENTAJA DE SOLARIS
Sun WebServer:SPEC-web96 (medida de la habilidad de un sistema para responder a una carga pesada de HTTP)	Da servicio a un mayor número de usuarios.	1322 SPEC-web96HTTP ops/s	3746 SPEC-web96HTTP ops/s	3 veces mejor rendimiento.
CaffeineMark 2.5 (medida de la velocidad de ejecución de Java y los gráficos 2-D simples)	Rendimiento alto en las aplicaciones Java.	5611	13950	2.5 mejor rendimiento.
Javac (indica el rendimiento de aplicaciones grandes que requieren un rápido acceso a datos)	Rendimiento alto en las aplicaciones Java.	451	57	8 veces un accesos más rápido.
Número máximo de usuarios que una configuración típica puede manejar		100	1000	Un servidor Solaris puede dar el servicio de vários NT.
Máxima memoria física de apoyo.		4 GB	30 GB	8 veces más memoria.
Habilidad para manejar múltiples usuarios locales y remotos en el mismo sistema.		No	Si	Mayor flexibilidad.

diseño mientras que Unix se encuentra en desarrollo en diversos frentes.

En todo caso un entorno mixto puede ser la solución a todos nuestros problemas. Aunque son dos sistemas bastante distintos existen varias formas de hacer que convivan juntos y sacarles el mayor rendimiento a cada uno de ellos. Por medio de la integración podemos aprovechar los puntos fuertes de cada uno para satisfacer nuestras necesidades. Podemos compartir archivos exportados mediante NFS (sistema de archivos distribuido al que se pueden acceder independientemente del sistema operativo utilizado) desde un NT o mediante Samba, el cual permite que los recursos compartidos por estaciones Unix puedan ser empleados por ordenadores que funcionan bajo Windows. Mediante Samba también podemos acceder y transmitir archivos a recursos compartidos en equipos Windows.

BIBLIOGRAFÍA

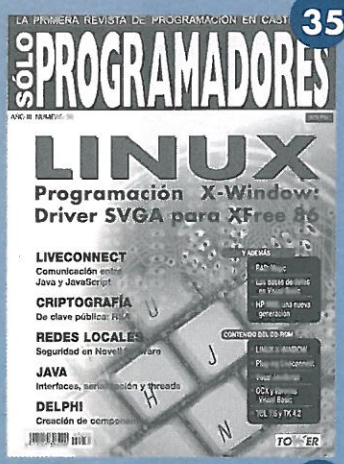
A continuación os facilitamos algunas de las direcciones que os pueden ayudar con respecto a este tema:

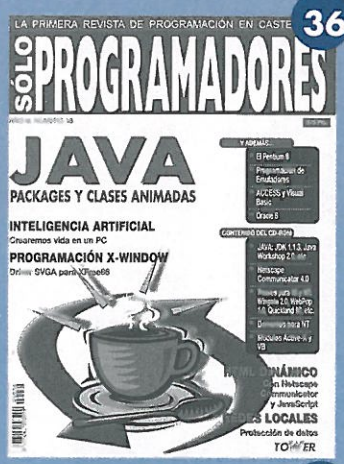
www.microsoft.com
www.sun.com
www.fisica.uson.mx
http://www.unix.digital.com/unix/v4/dhbrown/AIX43.htm

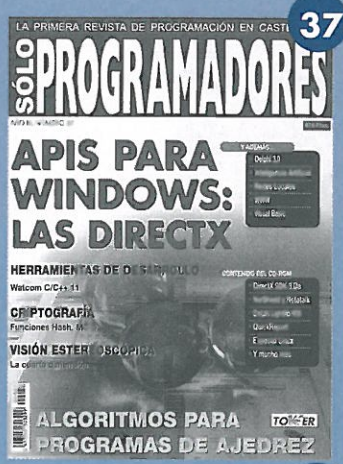
números atrasados

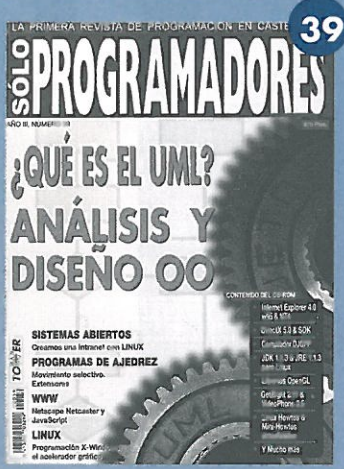
SÓLO PROGRAMADORES

completa ya tu colección









suscríbete

a Sólo Programadores

y consigue un magnífico descuento

suscripción normal

ahorro **20%**

12 revistas (1 año) por sólo...

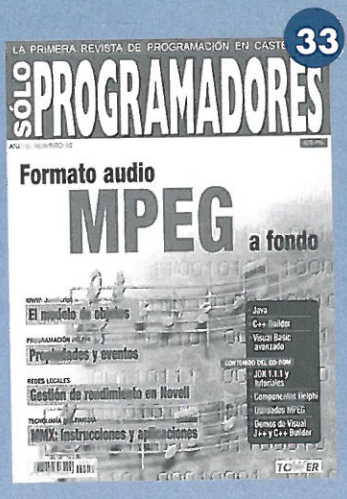
9.350 ptas.

suscripción estudiantes
(carreras técnicas)

ahorro **40%**

12 revistas (1 año) por sólo...

7.050 ptas.



Reconocimiento de voz (III)

Constantino Sánchez Ballesteros (constantino@redestb.es)

Como se prometió en el artículo anterior de la serie, vamos a abordar el primer proyecto de reconocimiento de comandos por medio de la API Speech. Controlaremos dos opciones del menú del procesador de textos Microsoft Word mediante los comandos que dictemos (menú Archivo y menú Edición).

Para poder controlar otro programa desde el nuestro, necesitamos crear una aplicación con soporte *DDE*. La *API Speech* no puede hacerlo por sí misma puesto que no tiene el soporte necesario para efectuar este tipo de tareas. En realidad, sólo se encarga de informarnos del estado del habla por parte del usuario para que, seguidamente, hagamos el trabajo necesario con los datos obtenidos mediante los mensajes de voz descifrados.

String especifica las pulsaciones de teclas que se van a enviar

Una aplicación con soporte *DDE* es capaz de establecer comunicación directa con otra aplicación para transferir datos y efectuar tareas según los datos obtenidos. Visual Basic posee varias instrucciones que trabajan directamente con la arquitectura *DDE*, y será una de ellas la que utilizaremos para enviar los comandos de voz a los demás programas: *SendKeys*.

SENDKEYS

Se encarga de enviar una o más pulsaciones de teclas a la ventana activa, como si se hubieran presionado desde el teclado. Cuando se reciban comandos hablados por parte del usuario, enviaremos las teclas necesarias a la aplicación destino según el comando que se haya dicho por el micrófono. Por ejemplo, si queremos activar el menú archivo de Microsoft Word, esperaremos a que el usuario pronuncie la palabra "archivo" o "menú archivo" y enviaremos las teclas ALT+A para activar la opción de menú en Word.

SINTAXIS DE LA INSTRUCCIÓN

```
SendKeys string[, wait]
```

La sintaxis de la instrucción **SendKeys** consta de los siguiente argumentos:

Parte Descripción

String: Expresión de cadena que especifica las pulsaciones de teclas que se van a enviar. (Requerido).

Wait: Valor Booleano que indica el modo de espera. Si este valor es **False** (predeterminado), se devuelve el control al procedimiento inmediatamente después de enviar las pulsaciones. Si es **True**, se procesan las pulsaciones antes de devolver el control al procedimiento. (Opcional).

Mediante el soporte DDE, controlaremos diversos programas simultáneamente utilizando el reconocimiento de voz

COMENTARIOS

Cada tecla está representada por uno o más caracteres, de tal forma que para especificar un único carácter del teclado, se utiliza el carácter propia-

Tabla 1. Caracteres que no se muestran al presionar una tecla.

Tecla	Código
RETROCESO	{BACKSPACE}, {BS}o {BKSP}
INTER	{BREAK}
BLOQ MAYÚS	{CAPSLOCK}
SUPR	{DELETE} o {DEL}
CURSOR ABAJO	{DOWN}
FIN	{END}
ENTRAR	{ENTER}o ~
ESC	{ESC}
AYUDA	{HELP}
INICIO	{HOME}
INSERT	{INSERT} o {INS}
CURSOR IZQRDA	{LEFT}
BLOQ NÚM	{NUMLOCK}
RE PÁG	{PGDN}
AV PÁG	{PGUP}
IMPR PANT	{PRTSC}
CURSOR DRCHA	{RIGHT}
BLOQ DESPL	{SCROLLLOCK}
TAB	{TAB}
CURSOR ARRIBA	{UP}
F1	{F1}
F2	{F2}
F3	{F3}
F4	{F4}
F5	{F5}
F6	{F6}
F7	{F7}
F8	{F8}
F9	{F9}
F10	{F10}
F11	{F11}
F12	{F12}
F13	{F13}
F14	{F14}
F15	{F15}
F16	{F16}

mente dicho. Por ejemplo, para representar la letra *A*, utilizaremos "A" como *string*. Si deseamos representar más de un carácter, agregaremos cada carácter adicional al que lo precede. Por ejemplo, para representar las letras A, B y C, utilizaremos "ABC".

El signo más (+), el símbolo de intercalación (^), el símbolo de porcentaje (%), la tilde (~) y los paréntesis ( ) tienen significados especiales para *SendKeys*. Aunque los corchetes ([ ]) no tienen un significado especial para *SendKeys* se deben escribir entre llaves debido a que, en otras aplicaciones los tienen y resulta de gran importancia cuando se produce un intercambio dinámico de datos (*DDE*). Para especificar llaves, utilizaremos {{} y {}}.

Para especificar caracteres que no se muestran al presionar una tecla (como por ejemplo *ENTRAR* o *TAB*) y teclas que representan acciones, en lugar de caracteres se utilizarán los códigos de tabla 1.

Una aplicación con soporte DDE es capaz de establecer comunicación directa con otra aplicación para transferir datos y efectuar tareas según los resultados obtenidos

Para especificar teclas con cualquier combinación de las teclas *MAYUS*, *CTRL* y *ALT*, introduciremos delante del código de la tecla uno o más de los siguientes códigos:

Tecla	Código
MAYÚS	+
CTRL (CONTROL)	^
ALT	%

Para especificar que se debe mantener presionada una combinación cualquiera de las teclas *MAYUS*, *CTRL* y *ALT* mientras se presionan otras teclas, debemos poner entre paréntesis el código relativo a esas teclas. Por ejemplo, para determinar que se debe mantener presionada *MAYUS* mientras se presionan las teclas *E* y *C*, utilizaremos "+(EC)". Para especificar que se debe mantener presionada *MAYUS* mientras se presiona la tecla *E* y que luego se debe presionar *C* sin presionar *MAYUS*, utilizaremos "+EC".

El menú de voz se encargará de contener todos aquellos comandos de voz que el usuario puede decir

Para especificar pulsaciones repetidas de teclas, utilizaremos el formato {*tecla número*}. Debemos dejar un espacio en blanco entre *tecla* y *número*. Por ejemplo, {**LEFT 42**} significa que se debe presionar la tecla *CURSOR IZQUIERDO* **42** veces; {**h 10**} significa que se debe presionar la tecla *h* **10** veces.

Nota: No puede utilizar *SendKeys* para enviar pulsaciones de teclas a aplicaciones que no estén diseñadas para funcionar con Microsoft Windows. *SendKeys* puede enviar también la pulsación de la tecla *IMPR PANT* (*PRTSC*) a cualquier aplicación.

CONSTRUCCIÓN DEL PROYECTO

Para la creación del proyecto se utilizarán los siguientes módulos y formularios:

Formulario **Form1**: Es el encargado de representar la ventana en la cual se informará del estado del reconocimiento de voz. Si se dicta un comando reconocido, aparecerá en el formulario; en caso contrario, aparecerá un mensaje que informará al usuario que no se ha reconocido el comando hablado.

Form 1 se encarga de representar la ventana en la cual se informará del estado del reconocimiento de voz

Módulo **Variables**: La tarea de este módulo consiste en definir las variables que representarán a los objetos "comandos de voz" y "menú de voz".

Módulo de clase **VcmdSink**: Consiste en definir las funciones *callbacks* para el reconocimiento de los comandos hablados.

A la hora de construir el proyecto, es necesario agregar en la opción **Referencias** del menú **Proyecto** de Visual Basic las librerías encargadas de gestionar la *API Speech*:

- VoiceCommand 1.0 Type library
- VoiceText 1.0 Type library
- Dictctl OLE Control Module

En la figura 2 se puede apreciar el detalle de la inclusión de estas librerías en el proyecto.

No es necesario tener en primer plano nuestra aplicación de reconocimiento para obtener los comandos hablados

Nota: Estas librerías son agregadas al disco duro cuando se instala la *API Speech* de Microsoft.

MÓDULO DE CLASE VARIABLES

Vamos a dar comienzo a la programación por este módulo, y en primer lugar tendremos que encargarnos de definir los dos objetos principales que componen el sistema de reconocimiento de voz:

- Un objeto "comandos de voz" para gestionar los comandos hablados por el usuario.
- Un objeto "menú de voz" para delimitar qué comandos se pueden decir.

```
Public VCmd As VCmdAuto.VCmdAuto
Public vmenu As VCmdAuto.IVMenuAuto
```

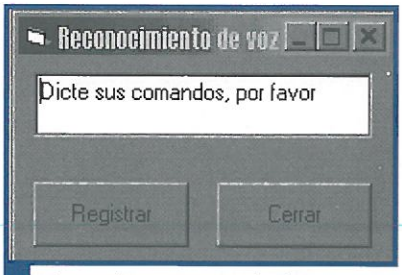


Figura 1. Ventana principal del proyecto de reconocimiento.

En el siguiente procedimiento se designa como formulario principal **Form1**:

```
Sub Main()
Form1.Show
End Sub
```

MÓDULO DE CLASE VCMDSINK

Como se comentó anteriormente, este módulo de clase gestiona las funciones *callback* del reconocimiento. En la primera función (**CommandOther**) se gestionan los comandos del usuario que no se han reconocido. Puede ser debido a una mala colocación del micrófono o que simplemente no existan en el menú de voz que hemos creado (formulario **Form1**).

La instrucción AppActivate define el nombre de la aplicación que vamos a controlar

Si la cadena de caracteres obtenida en la variable *pszCommand* es cero, significa que el comando actual no se ha reconocido y se visualizará en el cuadro de texto del formulario un mensaje de **REPITA, POR FAVOR**. De este modo, el usuario sabrá si su comando se ha reconocido correctamente.

```
Function CommandOther(pszCommand As
      String, pszApp As String,
                  pszState As String)
```

```
If Len(pszCommand) = 0 Then
  Form1.Text1.Text = "REPITA, POR
    FAVOR"
  End If
End Function
```

En la siguiente función se gestionan las palabras que han sido reconocidas por el *engine* de reconocimiento:

Function CommandRecognize(pszCommand As String, dwID As Long)

Siempre es necesario que registremos nuestro programa al engine de reconocimiento con el que trabajamos

En la variable *Form1* que representa al formulario se asignará la cadena de caracteres que representa al comando reconocido por el *engine*:

```
Form1.Text1.Text = pszCommand
```

En primer lugar debemos definir las condicionales que determinan el comando que reconocer dentro del menú *Archivo* de Microsoft Word. Mediante la instrucción *AppActivate* definimos el nombre de la aplicación que vamos a controlar, teniendo en cuenta que este nombre aparecerá en la parte superior izquierda de la ventana de la aplicación (en nuestro caso Microsoft Word).

En la instrucción *SendKeys* asignaremos los parámetros que determinan la pulsación de teclas para cada opción del menú. Puesto que para activar dicho menú se utiliza la tecla *ALT*, definiremos el carácter % (véase la tabla anterior relativa a *SendKeys*). Seguidamente pondremos entre paréntesis las teclas que deben pulsarse para llegar hasta la opción correspondiente dentro del menú.

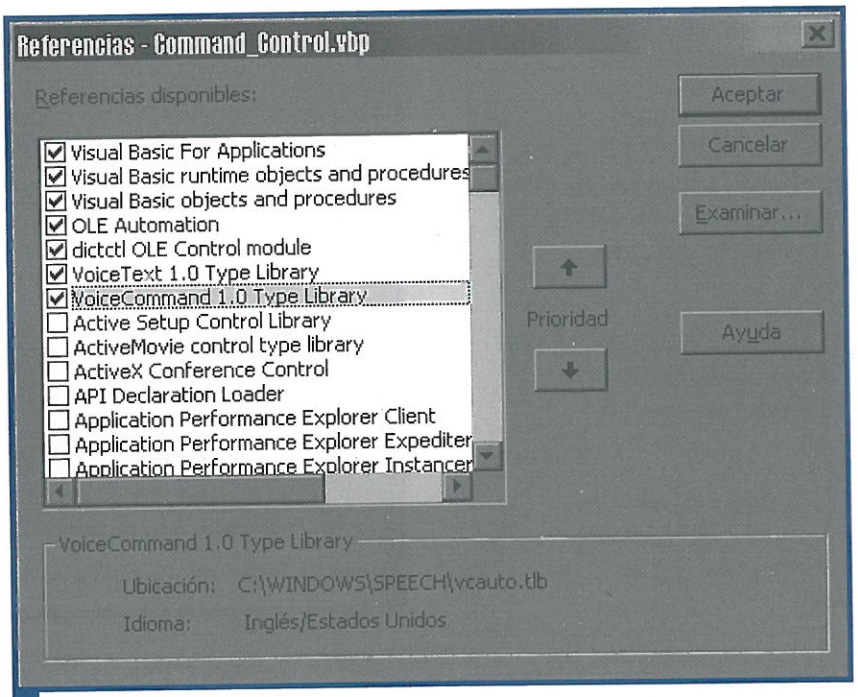


Figura 2. Detalle de la inclusión de librerías de la API Speech.

La variable *pszCommand* debe igualarse a cualquiera de los comandos que hayamos introducido en el menú de voz (véase más adelante en la programación del formulario *Form1*). Primero definimos las condicionales del menú *Archivo* y posteriormente las del menú *Edición*.

En la variable Form 1 que representa al formulario, se asignará la cadena de caracteres que del comando reconocido por el engine

Nota: las demás opciones del menú de Word no se representan en estas páginas por motivos de espacio. Su diseño sería análogo: definir dentro de esta función todas las opciones posibles del menú y agregarlas al "menú de voz" (definido en el formulario *Form1*).

Pasamos a partir de este mismo momento a detallar la definición que corresponde al menú *Archivo* y sus opciones:

```
If pszCommand = "Nuevo" Then
  AppActivate "Microsoft Word"
  SendKeys "%(a)+(n)", True
End If

If pszCommand = "Abrir" Then
   AppActivate "Microsoft Word"
   SendKeys "%(a)+(a)", True
End If

If pszCommand = "Cerrar" Then
   AppActivate "Microsoft Word"
   SendKeys "%(a)+(c)", True
End If

If pszCommand = "Guardar" Then
   AppActivate "Microsoft Word"
   SendKeys "%(a)+(g)", True
End If

If pszCommand = "Guardar como" Then
   AppActivate "Microsoft Word"
   SendKeys "%(a)+(u)", True
End If
```

```
If pszCommand = "Versiones" Then
   AppActivate "Microsoft Word"
   SendKeys "%(a)+(e)", True
End If

If pszCommand = "Configurar página" Then
   AppActivate "Microsoft Word"
   SendKeys "%(a)+(r)", True
End If

If pszCommand = "Vista preliminar" Then
   AppActivate "Microsoft Word"
   SendKeys "%(a)+(m)", True
End If

If pszCommand = "Imprimir" Then
   AppActivate "Microsoft Word"
   SendKeys "%(a)+(i)", True
End If

If pszCommand = "Enviar a" Then
   AppActivate "Microsoft Word"
   SendKeys "%(a)+(v)", True
End If

If pszCommand = "Propiedades" Then
   AppActivate "Microsoft Word"
   SendKeys "%(a)+(p)", True
End If

If pszCommand = "Salir" Then
   AppActivate "Microsoft Word"
   SendKeys "%(a)+(s)", True
End If
```

Cuando se agregan nuevos comandos al menú de voz, hemos de asegurarnos que las palabras incluidas están en el diccionario del engine

Definición del menú *Edición* y sus opciones:

```
If pszCommand = "Deshacer" Then
   AppActivate "Microsoft Word"
   SendKeys "%(e)+(d)", True
End If

If pszCommand = "Repetir" Then
   AppActivate "Microsoft Word"
   SendKeys "%(e)+(r)", True
End If

If pszCommand = "Cortar" Then
   AppActivate "Microsoft Word"
   SendKeys "%(e)+(t)", True
End If

If pszCommand = "Copiar" Then
   AppActivate "Microsoft Word"
   SendKeys "%(e)+(c)", True
End If

If pszCommand = "Pegar" Then
   AppActivate "Microsoft Word"
   SendKeys "%(e)+(p)", True
End If

If pszCommand = "Pegado especial" Then
   AppActivate "Microsoft Word"
   SendKeys "%(e)+(e)", True
End If

If pszCommand = "Pegar como hipervínculo"
      Then
   AppActivate "Microsoft Word"
   SendKeys "%(e)+(m)", True
End If

If pszCommand = "Borrar" Then
   AppActivate "Microsoft Word"
   SendKeys "%(e)+(a)", True
End If

If pszCommand = "Seleccionar todo" Then
   AppActivate "Microsoft Word"
   SendKeys "%(e)+(s)", True
End If

If pszCommand = "Buscar" Then
   AppActivate "Microsoft Word"
   SendKeys "%(e)+(v)", True
End If

If pszCommand = "Reemplazar" Then
   AppActivate "Microsoft Word"
   SendKeys "%(e)+(z)", True
End If

If pszCommand = "Ir a" Then
   AppActivate "Microsoft Word"
   SendKeys "%(e)+(i)", True
End If

If pszCommand = "Vínculos" Then
   AppActivate "Microsoft Word"
   SendKeys "%(e)+(v)", True
End If

If pszCommand = "Objeto" Then
   AppActivate "Microsoft Word"
   SendKeys "%(e)+(o)", True
End If

End Function
```

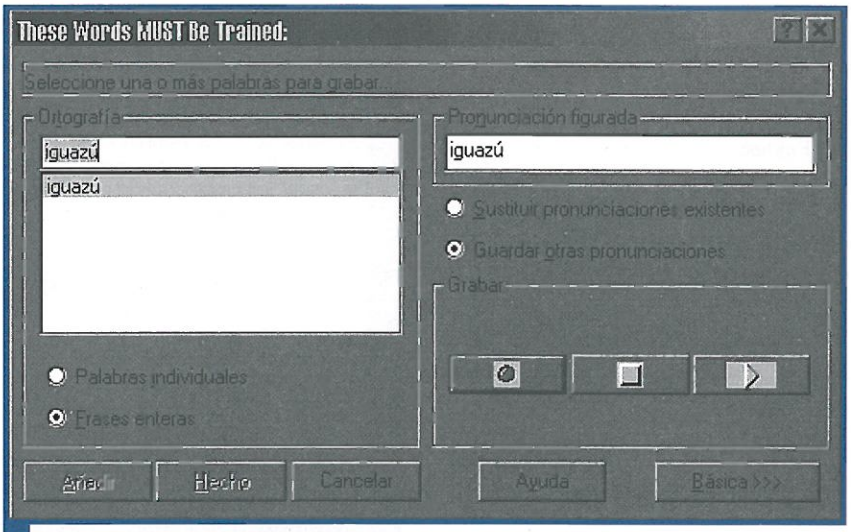


Figura 3. Entrenamiento de palabras no reconocidas en el menú.

FORMULARIO FORM1:

El siguiente procedimiento representa al botón **Registrar** del formulario que se encarga de registrar la aplicación en el *engine* de reconocimiento. Esta operación resulta indispensable para inicializar la *API*.

Nota: se puede registrar la aplicación de forma automática sin necesidad de pulsar un botón si definimos un procedimiento que se ejecute al cargar el formulario principal.

```
Private Sub Command1_Click()
```

Lo primero que debemos hacer es borrar el objeto "comandos de voz" (*VCmd*) antes de efectuar el registro. Seguidamente, crearemos el objeto mediante la función *CreateObject*.

```
Set VCmd = Nothing
Set VCmd = CreateObject("Speech.VoiceCommand")
```

Registramos la aplicación:

```
Call VCmd.Register("")
```

Definimos la función *callback* establecida en el módulo de clase:

```
VCmd.Callback = "comandos.VCmdSink"
```

Creamos un menú de voz para poder agregarle los comandos que reconocer:

```
Set vmenu = VCmd.MenuCreate("Comandos de voz", "prueba", 0, "español", vcmdmc_CREATE_ALWAYS)
```

Activamos el micrófono para poder recibir los comandos audibles del usuario:

```
VCmd.Awake = True
```

Introducimos los comandos del menú de Word (*Archivo* y *Edición*) en el menú de voz creado:

```
Call vmenu.Add(1, "Nuevo", "general", "prueba")
Call vmenu.Add(2, "Abrir", "general", "prueba")
Call vmenu.Add(3, "Cerrar", "general", "prueba")
Call vmenu.Add(4, "Guardar", "general", "prueba")
Call vmenu.Add(5, "Guardar como", "general", "prueba")
Call vmenu.Add(6, "Versiones", "general", "prueba")
Call vmenu.Add(7, "Configurar página", "general", "prueba")
Call vmenu.Add(8, "Vista preliminar", "general", "prueba")
Call vmenu.Add(9, "Imprimir", "general", "prueba")
Call vmenu.Add(10, "Enviar a", "general", "prueba")
Call vmenu.Add(11, "Propiedades", "general", "prueba")
Call vmenu.Add(12, "Salir", "general", "prueba")

Call vmenu.Add(13, "Deshacer", "general", "prueba")
Call vmenu.Add(14, "Repetir", "general", "prueba")
Call vmenu.Add(15, "Cortar", "general", "prueba")
Call vmenu.Add(16, "Copiar", "general", "prueba")
Call vmenu.Add(17, "Pegar", "general", "prueba")
Call vmenu.Add(18, "Pegado especial", "general", "prueba")
Call vmenu.Add(19, "Pegar como hipervínculo", "general", "prueba")
Call vmenu.Add(20, "Borrar", "general", "prueba")
Call vmenu.Add(21, "Seleccionar todo", "general", "prueba")
Call vmenu.Add(22, "Buscar", "general", "prueba")
Call vmenu.Add(23, "Reemplazar", "general", "prueba")
Call vmenu.Add(24, "Ir a", "general", "prueba")
Call vmenu.Add(25, "Vínculos", "general", "prueba")
Call vmenu.Add(26, "Objeto", "general", "prueba")
```

Y por último activamos el menú:

```
vmenu.Active = True

End Sub
```

Este procedimiento es el que se emcarga de representra al botón **Cerrar** del formulario y se encarga de finalizar la aplicación.

```
Private Sub Command2_Click()
End
End Sub
```

Cuando se descargue el formulario, debemos eliminar los objetos de la *API* creados (Comandos de voz y menú de voz):

```
Private Sub Form_Unload(Cancel As Integer)
  Set VCmd = Nothing
  Set vmenu = Nothing
  End
End Sub
```

Cuando se agregan nuevos comandos al menú de voz, debemos asegurarnos de que las palabras incluidas se encuentran en el diccionario del *engine*. De lo contrario, al registrar el menú se mostrará un formulario como el siguiente invitándonos a registrar la nueva palabra en el diccionario:

> Lo primero que debemos hacer es borrar el objeto "comandos de voz" (VCnd) antes de efectuar el registro, después lo volveremos a crear

Para que el reconocimiento de voz funcione con Word, este último debe estar encontrarse en ejecución. También se podría utilizar el método *Shell* de Visual Basic para poder ejecutar automáticamente el procesador de textos cuando se inicie nuestra aplicación de reconocimiento.

Protección de servidores y redes en Internet (I)

Juan José Taboada León (taboada@uhu.es) y José Juan Mora Pérez (albeniz@uhu.es)

Actualmente todo el mundo, tanto usuarios particulares, como grandes empresas utilizan los servicios de Internet, pero muchas veces no somos conscientes de la información que inunda la red y de los problemas que ello puede acarrear. ¿Existe seguridad para todos los datos que se manejan?

Internet es uno de los medios de comunicación más populares hoy en día, todo el mundo quiere conectarse a la red, desde las grandes empresas hasta cualquier persona que nada tiene que ver con la informática. Esas mismas empresas han visto en Internet otra forma más de vender, pero con la diferencia, que en Internet, el número de clientes potenciales se multiplica cada día y las infraestructuras necesarias para dar a conocer sus productos a los clientes ya no son los caros anuncios de televisión, las páginas en una revista o las vallas publicitarias.

Con Internet, únicamente necesitan una página web y tantos *banners* como sea necesario. De igual forma, los usuarios pueden aprovechar las múltiples formas de comunicación que ofrece esta red, desde el simple correo electrónico, pasando por el IRC o las vídeoconferencias. Se habla de *"la aldea global"* y *"la comunicación total"*, donde cualquiera puede decir, vender o comprar cualquier cosa.

Sociológicamente no podemos dudar de la importancia de Internet, pero no debemos olvidar qué es en realidad Internet. Se trata de una red de ordenadores, la cual se utiliza para transmitir INFORMACIÓN, (esta es la palabra clave), haciendo posible el trabajo con la misma de una forma que ningún otro medio ha conseguido hasta ahora. Esta es la verdadera causa de su espectacular crecimiento.

Al abrigo de este crecimiento, ha surgido un problema importante, ya que nos encontramos ante uno de los medios más peligrosos para trabajar con la información. Pongamos un ejemplo: deseamos realizar una compra utilizando alguna de las múltiples páginas disponibles en la web, y una vez hemos decidido qué comprar, nos solicitan los datos personales, así como los datos bancarios necesarios para realizar la compra, normalmente utilizando alguna tarjeta de crédito. Si la empresa en la que estamos realizando la compra no utiliza algún sistema de seguridad, nuestros datos bancarios estarán circulando por la red de una forma totalmente insegura, y no tenemos conocimiento que estén pasando por un ordenador seguro o inseguro, es decir no sabemos si nuestros datos han sido interceptados, con algún programa del tipo *Sniffer*. Estos programas capturan toda la información que pasa por la red en la que están escuchando (como puede ser **Tcpdump**), pudiendo modificar y volver a transmitir nuestros datos, con lo que alguien en algún sitio puede estar son-

riendo mientras mira en la pantalla los números de nuestra tarjeta de crédito.

PRIVACIDAD DE LOS DATOS

La seguridad en Internet y por lo tanto la seguridad en los servidores de información que conforman esta red, es uno de los campos en los que más medios se están invirtiendo. De nada vale contar con el llamado *"medio de comunicación total"*, si dicho medio es inseguro y no permite realizar transferencias de información de una forma fiable.

Desde un punto de vista general la seguridad informática se centra en dos amplias parcelas, una parcela técnica, consistente en una larga lista de herramientas tanto software como hardware, y una segunda donde debemos considerar el factor humano, es decir el conjunto de personas encargadas de implantar el sistema de seguridad, administradores y todas aquellas personas que utilizarán consciente o inconscientemente dicho sistema. No obstante son tantos los aspectos que considerar dentro de la seguridad, que no existe una herramienta universal que proteja la integridad de los datos con los que trabaja un sistema informático. El conjunto de herramientas y la experiencia de los administradores, conforman una ecuación en la cual la experiencia del administrador tienen el mayor peso específico.

Todo sistema de seguridad que implantemos debe cumplir dos requisitos fundamentales: por un lado, proporcionar **confidencialidad** para evitar que personas no autorizadas tenga acceso a los datos y por otro ofrecer **integridad** para asegurar que dicha información sea correcta y no haya sido manipulada por personas no autorizadas. Para cumplir estos dos requisitos existen numerosas herramientas y estrategias como la encriptación de la información, la utilización de canales de comunicación confiables, la implantación de esquemas de seguridad para proteger los servidores de información o las redes de las organizaciones conectadas a Internet, etc. La implantación de un sistema de seguridad basado en la encriptación de la información, estará supeditado a los algoritmos y herramientas que decidamos utilizar.

Podemos enumerar algunos:

- DES
- DEA
- RC2 y RC4
- RSA
- PGP

Uno de los mayores problemas de seguridad en cualquier sistema son las palabras de paso (*passwords*) que utilizan los usuarios. Existen herramientas que permiten disminuir el riesgo de que dichas palabras de paso sean fáciles de adivinar. Otra estrategia es utilizar un sistema como puede ser *S/KEY* u *OPIE* (*One-Time Passwords In Everything*), los cuales trabajan con palabras de paso que únicamente se utilizarán una vez y por lo tanto evitamos el problema de que una palabra de paso interceptada, sea utilizada para intentar entrar en un sistema.

En cuanto al establecimiento de canales confiables de comunicación, podemos citar herramientas tan populares como *SSH* (*Secure Shell*) o *SSL* (*Secure Socket Layer*). Ambas permiten establecer una conexión totalmente segura con una máquina remota. *SSH* reemplaza a las conocidas utilidades **rsh**, **rcp** y **rlogin**, las cuales trabajan con texto llano, es decir, la información circula a través de una conexión si ningún tipo de encriptación, por herramientas que emplean alguno de los algoritmos de encriptación citados. *SSL* ha sido desarrollado por la empresa Netscape y se encuentra integrado en sus navegadores, de tal forma que cuando accedemos a algún servidor *HTTP* que soporte canales seguros, el icono que representa una llave rota o candado abierto, se convertirá en una llave entera (candado cerrado), lo que indica que se está utilizando un canal seguro de comunicación.

El objeto de este artículo es mostrar cuáles son los esquemas de seguridad existentes para proteger servidores de información y redes. Actualmente estos esquemas de seguridad se sustentan en lo que se ha convenido en llamar **firewall** o **cortafuego**. Veremos en qué consisten y con qué herramientas contamos a la hora de diseñarlos, dejando para artículos posteriores, el análisis de herramientas concretas como *OPIE*, *SATAN* y *SSH*. De esta forma, realizaremos un recorrido por las distintas herramientas de seguridad que se deben tener en cuenta a la hora de plantear un esquema de este tipo.

Antes de entrar en materia se indicarán algunos de los aspectos básicos que se han de contemplar a la hora de planificar la seguridad de cualquier servidor. El pilar que soporta todo el peso de nuestro sistema de seguridad es la política de seguridad que nosotros como administradores deseemos o en algunos casos estemos obligados a implantar. De nada sirve tener los mejores equipos del mundo si nuestra política de seguridad permite que los usuarios tengan, por ejemplo, palabras de paso de tres letras, las cuales son fáciles de adivinar.

- Eliminar todas las cuentas de usuarios innecesarias. Esto es bastante importante, ya que uno de los métodos más utilizados para entrar ilegalmente en un sistema, es la utilización de alguno de los numerosos programas que existen para adivinar claves, como

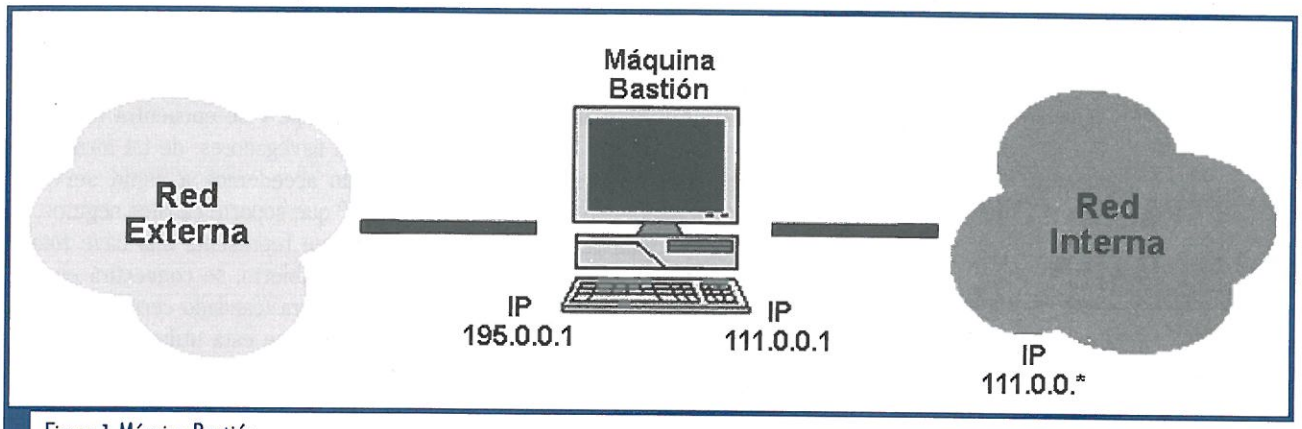


Figura 1. Máquina Bastión.

es el caso de **Crack**. Su funcionamiento se base en el empleo de un diccionario, mediante el cual busca las posibles palabras de paso que puede tener una cuenta de usuario. Cuantas menos cuentas tenga nuestro sistema, menor será la probabilidad de que alguna de ellas tenga una palabra de paso débil, fácil de adivinar.

- Eliminar todos los compiladores, linkadores, fuentes de programas, etc. Esto puede parecer poco práctico, sobre todo en sistemas UNIX, donde la mayoría del software debe ser compilado, pero es una buena política el que el cortafuego no cuente con ninguna de estas herramientas. Esto evitará que el intruso pueda crear un programa o modificar el sistema operativo sobre todo en sistemas LINUX, que se distribuyen con las fuentes. Ha habido casos en los cuales, los atacantes, una vez dentro de una máquina, han construido versiones de los comandos **ps** y **ls** para que no delatasen ficheros y procesos utilizados para obtener contraseñas o cualquier otra información delicada de la máquina.

- Eliminar los servicios que no estén siendo utilizados en la máquina, así como evitar tener abiertos puertos que no son necesarios. Por ejemplo, si nuestra máquina no trabaja con correo electrónico no es necesario tener abierto el puerto de correo (puerto 25); sería una excelente idea cerrar el puerto de *finger* (puerto 79).

- Eliminar el acceso remoto a la máquina como usuario **root**, para que dicho acceso únicamente pueda realizarse a través de la consola.

- Realizar auditorias periódicamente. Es importante que regularmente se revise el sistema para comprobar su rendimiento, así como actualizar los posibles fallos de seguridad que se vayan descubriendo. Un sistema seguro hoy, no tiene por qué ser seguro mañana. Día a día se van descubriendo nuevos *bugs* en la seguridad de los sistemas. Es una obligación, que nosotros, como administradores, estemos informados de todos estos fallos para poder corregirlos. Una de las herramientas más utilizadas para detectar posibles fallos de seguridad en nuestro sistema es *SATAN* (*Security Administrator´s Tool for Analyzing Networks*), en un artículo posterior estudiaremos esta herramienta con profundidad.

- Los registros de acceso deben examinarse para buscar posibles intentos de ataques. Esta es una de las labores más importantes de un administrador, ya que si nuestro cortafuego es atacado, pero lo detectamos a tiempo, podemos evitar males mayores. En los ficheros de acceso se registran todos los accesos que se realizan a nuestra máquina y también los intentos para acceder. Analizando este fichero podríamos comprobar si alguien está intentando entrar repetidamente desde un mismo sitio y de esta forma poder actuar en consecuencia. Para este trabajo es necesario que conozcamos cómo trabaja y se configura uno de los demonios de los sistemas UNIX, **syslogd**, que registra todos los sucesos correspondientes a la autentificación de usuarios, etc.

¿QUÉ ES REALMENTE UN CORTAFUEGO?

Sobre el concepto de cortafuego existe cierta polémica, ya que no todo el mundo se refiere a lo mismo cuando se hace mención al mismo. En nuestro caso entendemos como cortafuego todo elemento que evita que el fuego se propague, como por ejemplo, las veredas forestales, las cuales evitan, a veces,

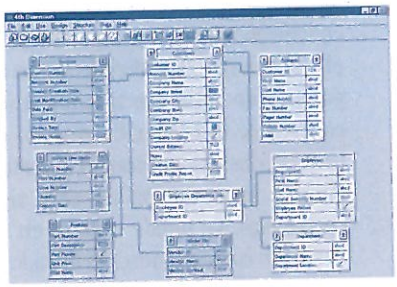
Si le gustaron otras bases de datos, 4th Dimension v6 *le sorprenderá*

4th Dimension es el entorno de bases de datos integrado más completo del mercado. Años de experiencia en el diseño de bases de datos han permitido satisfacer las siguientes necesidades:

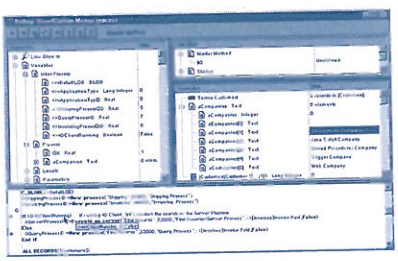
- Automatizar cualquier proceso.
- Diseñar modelos gráficos eficientes y orientados a objeto.
- Suministrar la más amplia gama de herramientas para facilitar el diseño y mantenimiento de bases de datos.
- Permitir una fácil reutilización de código.
- Permitir escalar desarrollos Cliente/Servidor, web y multiplataforma de forma transparente.

Estas son algunas de las **características** de 4th Dimension:

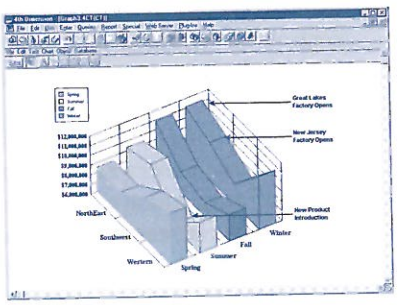
- **Motor de base de datos multiproceso, triggers, procedimientos almacenados**, presentación visual de tablas mediante diagrama de entidad /relación (ERD)



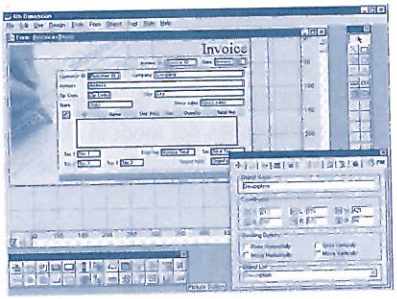
- **Debugger visual** multiproceso de última generación. **Intérprete** incluido para testeo y debugging.



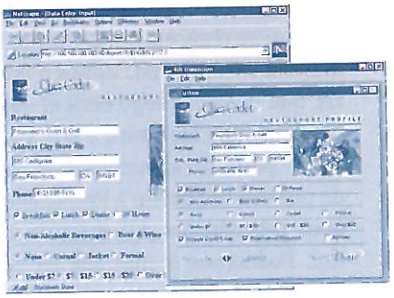
- **Wizards avanzados** para diseño de formularios, queries, informes, importación, exportación, etiquetas, etc. Editor de gráficas programable 2D y 3D, soporta gráficas de barras, tartas, áreas, líneas, etc.



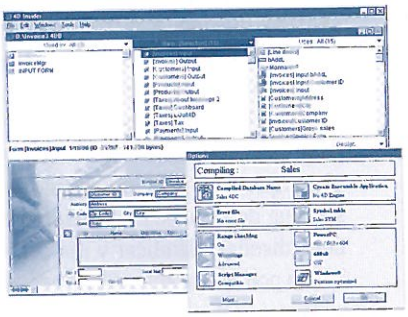
- **Editor de formularios basado en objetos.** Gestión de hojas de estilo. Redimensionamiento automático. **Docenas de objetos de interface** incluyendo check boxes, radio buttons, push buttons, áreas de scroll, picture button, botones invisibles, tab control, listas jerárquicas, termóme-tros, reglas, combo box, listas menú desplegables, menús pop-up jerárquicos, menús picture, radio pictures, listas desplegables, menús pop-up, subforms, group boxes, etc.



- Arquitectura de distribución de aplicaciones integrada y **soporte para proyectos con múltiples desarrolladores** con check-in/check-out automatizado.
- Arquitectura extensible mediante **plug-ins** y DLLs.
- **Un sólo código fuente** desplegable sin modificaciones de monousuario a 4D Server, la versión Cliente/Servidor de 4th Dimension.
- Soporte de **ODBC** (cliente y servidor) y conexiones nativas **SQL** a ORACLE, Sybase y MS SQL Server.
- **Servidor Web integrado** para acceso automático a formularios y gestión Cliente/Servidor mediante navegadores web.



- **Compilador a código máquina real** optimizado para diferentes procesadores. **Gestor gráfico de referencias cruzadas** y librerías de objetos.



VERSIÓN DEMO GRATUITA

4th Dimension es el núcleo de la gama de productos 4D. Visite http://www.ambitconsulting.com o llame al 93-209 41 11 para más información o para recibir una versión demo gratuita. Mencione este anuncio y reciba 4th Dimension v6 por el precio promocional de 34.900 Ptas*. (precio habitual: 79.000 Ptas.).

(* Actualización competitiva. Oferta válida hasta el final de existencias)

Balmes, 297, 4º, 2ºb. 08006 Barcelona
t. | (34) 93 209 41 11 f. | (34) 93 240 18 88
e-mail: ambit@ambitconsulting.com
http://www.ambitconsulting.com

AMBIT Consulting

que el fuego pase de un lado del bosque al otro lado. En el mundo de las redes de ordenadores, existe una similitud con el ejemplo de las veredas forestales, ya que podemos considerar que una parte del bosque sería nuestra red y la otra parte una red externa, para nosotros el fuego puede ser cualquier individuo que intente amenazar la integridad de nuestra red, ya sean *hackers*, *crackers*, empleados resentidos, etc., ver figura 1.

Por tanto, un cortafuego es todo elemento que proporciona un punto de defensa, control y auditoría de los accesos que se realicen a nuestra red desde una red externa. Así, vamos a considerar que un cortafuego será el conjunto de elementos hardware y software que van a permitir aumentar la seguridad de nuestra red frente a los posibles ataques procedentes del exterior o interior. Podríamos ir más allá en nuestra definición, e incluir en este conjunto, al elemento más importante de un cortafuego, el **administrador** de la red.

Siguiendo con el ejemplo de las veredas forestales, no tendría ningún sentido que se invirtiera dinero en construir dichas veredas, si en el momento en el que se produce un incendio los bomberos no aparecen para controlarlo, es decir el factor humano es el componente más importante de todo sistema de seguridad. El administrador debe diseñar, planificar y mantener una buena política de seguridad, y es en este punto, donde muchos de los sistemas de seguridad fallan de forma más que apreciable.

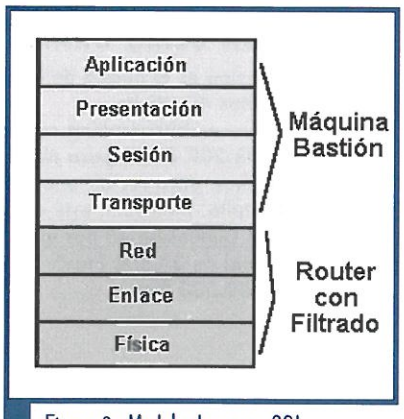


Figura 2 . Modelo de capas OSI.

Hace poco el mismísimo Pentágono reconoció públicamente que fue atacado por una oleada de *hackers*, lo que debe hacer reflexionar sobre la magnitud del problema al que nos enfrentamos. Internet se muestra como una red externa en la que no tendremos ningún control, siendo un medio totalmente hostil del cual tendremos que aislarnos de forma conveniente. La mejor arma con la que cuenta una persona que desee atacar nuestra red, ya sea mediante un ataque pasivo o con uno activo, es la astucia, aprovechándose de los fallos de seguridad de los sistemas. Es en este punto donde la experiencia de un administrador entra en juego, creando trampas, información falsa con la que entretener al intruso, teniendo puertas falsas, etc.

TIPOS DE CORTAFUEGOS

Básicamente podemos clasificar los cortafuegos en dos tipos, atendiendo a las capas del modelo OSI sobre las que actúa, tal y como se puede apreciar en la figura 2.

- Filtrado de paquetes.
- Máquina Bastión, la cual puede actuar también como servidor *proxy* de algún servicio.

Más adelante veremos la diferencia entre unos y otros.

Esta clasificación no es excluyente, es decir, el que tengamos una máquina con capacidad de filtrar paquetes, como puede ser un *router* dedicado, no significa que no podamos tener a la vez, una máquina Bastión que actúe como pasarela a nivel de aplicación (servidor *proxy*).

Ya hemos visto que cada uno abarca un conjunto de capas del modelo OSI, por lo tanto es obvio que si queremos diseñar un buen esquema de seguridad, el cual proteja nuestra red, éste debe contemplar el mayor número de niveles del modelo OSI. Más adelante veremos cómo podemos implementar un esquema de seguridad con ambos tipos de cortafuegos, para hacer de nuestra red, un lugar relativamente seguro.

PROTECCIÓN EN LA CAPA DE RED

Este tipo de cortafuego trabaja en la capa de red del modelo OSI, tal y como se muestra en la figura 2, y se basan en analizar la información que contienen los paquetes que llegan hasta el cortafuego, decidiendo si se dejan pasar o se rechazan basándose en su contenido, dirección de origen, dirección de destino, protocolo, puerto, etc. Para la implantación de este tipo de cortafuego, se puede utilizar un *router* dedicado con capacidad de filtrado o una máquina UNIX con capacidad de rutar tráfico. Un *router* dedicado es un elemento dedicado únicamente a encaminar el tráfico de una red y no dispone de ningún otro servicio. Antes de la aparición de estos elementos especializados eran las máquinas UNIX las encargadas de realizar este trabajo, pero hoy día lo realizan casi exclusivamente los *routers* dedicados, como son, los de la casa *CISCO System*, la marca más extendida en todo el mundo.

Por sí solo, un *router*, ya sea dedicado o una máquina UNIX, no es un elemento de seguridad, ya que se limita a retransmitir los paquetes que le lleguen. Pero si cuenta con la capacidad de filtrar

paquetes, es decir, se permite o deniega que ciertos paquetes atraviesen el *router*, según unos criterios que nosotros como administradores establezcamos, entonces contaremos con un elemento que nos ayudará a proteger nuestra red.

Como podemos ver en la figura 3, el *router* dispondrá de al menos dos interfaces, cada uno con una dirección IP, uno para la red interna y otro para la red externa. El proceso de filtrar los paquetes que llegan y salen del *router* se realiza mediante unas tablas de filtrado como podemos ver en las tablas que acompañan al artículo, en las cuales se intenta reflejar los criterios de selección de los paquetes. Habrá, al menos, una tabla que se aplicará a los paquetes que atraviesen la interfaz externa y otra tabla para los que lo hagan por la interfaz interna. Más adelante, en el apartado de *Configuraciones*, veremos para un caso concreto, cuál es la forma en la que debemos crear las reglas de filtrado, dependiendo de las restricciones que deseemos aplicar a nuestro sistema.

Una de las ventajas de este tipo de cortafuego es que si decidimos implantar un sistema de seguridad basado en el filtrado de paquetes, éste se realiza en el nivel de red, lo que significa que es totalmente transparente para los usuarios. De esta forma al no conocer la existencia del sistema de seguridad, no percibirán ningún tipo de molestia, frente al otro tipo de cortafuego, en los que los usuarios deben conocer su existencia y a veces modificar su forma de trabajo, para acomodarse a la forma en que trabaja el mismo.

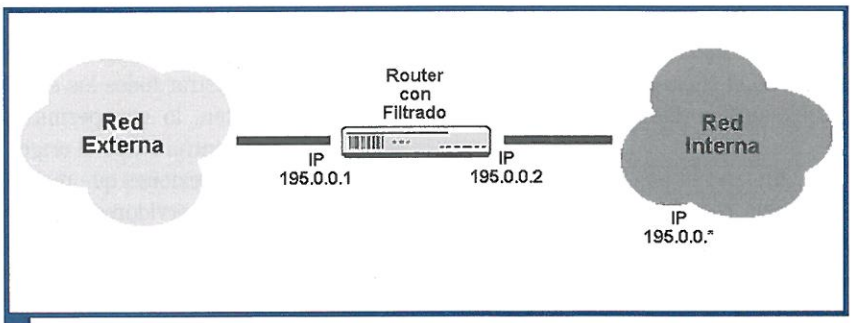


Figura 3. Filtrado de paquetes.

SERVIDOR BASTIÓN

Bastión es la palabra con la que se denomina a las estructuras mejor fortificadas de las que dispone un castillo para la defensa frente a un ataque. Por esta razón, se ha denominado máquina Bastión, a una máquina perteneciente a nuestra red, la cual va a estar fuertemente protegida y a la que deberán enfrentarse todos aquellos que intenten realizar un ataque sobre la red.

A diferencia del *router* dedicado con capacidad de filtrado, el servidor Bastión, actúa sobre las cuatro capas superiores de OSI. Por lo tanto, para un caso en el cual la máquina Bastión se encuentra entre la red interna y la red externa, los paquetes no circularán de una red a otra libremente, sino que todos se detendrán en esta máquina.

Con una máquina Bastión no es necesario que las maquinas de la red interna posean una IP real, podemos darles la IP que nos parezca, ya que ninguna de estas máquinas va a mandar paquetes fuera de la red, sino que dichos paquetes llegarán al Bastión y éste los encapsulará con la IP de la interfaz que da servicio a la red externa. Por lo tanto podemos dar servicio a una red con un solo número IP real, todos los demás serán ficticios.

Las máquinas Bastión serán el único punto de contacto con el exterior, que tendrá nuestra red, es decir todo el tráfico que circule desde nuestra red a la red externa y viceversa deberá ir dirigido a las distintas máquinas Bastión que protejan nuestra red. Nuestra política de seguridad no debería permitir que ninguna máquina de nuestra red, acceda al exterior directamente, sin pasar o por una máquina Bastión o por un *router* con filtrado.

Los sistemas UNIX cuentan con una opción, denominada de *IP Forwarding* que permite al sistema retransmitir todos los paquetes que lleguen por alguno de sus interfaces, con lo que actuaría como un *router*. Para construir una máquina Bastión, necesitamos que esta opción este deshabilitada ya que así evitaremos que los paquetes que lleguen al Bastión sean retransmitidos. De esta forma impediremos la comunicación entre la red interna y la red externa y por lo tanto para que una máquina de la red interna acceda a alguno de los servicios de la red externa, como por ejemplo un servidor FTP situado en la red externa, debería realizar un proceso similar al siguiente:

- Primero debe acceder al cortafuego mediante una conexión TELNET, lo cual indica que en el cortafuego deben existir cuentas de usuarios, lo que puede constituir una grieta importante en la seguridad del cortafuego y por lo tanto de nuestra red.

- Una vez que se ha establecido la conexión TELNET con el cortafuego tendremos que realizar una conexión FTP hacia la máquina situada en la red externa.

Como podemos deducir esta solución es poco atractiva para entornos en los cuales los usuarios de nuestra red no estén familiarizados con entornos

UNIX, ya que deberían realizar la transferencia de ficheros de una red a otra utilizando el cliente FTP que esté disponible en el cortafuego.

Otra opción podría consistir en la instalación de un servidor *proxy* en la máquina Bastión, con esto, conseguiríamos que los usuarios que deseen acceder a la red externa, no tengan que acceder primero al cortafuego, y únicamente tendrían que dirigirse al servidor *proxy* instalado en él, el cual, realizará las distintas peticiones a los servicios que el usuario solicite.

PROXY

El término inglés *proxy* puede traducirse al castellano como *delegado* o *procurador*. Estos servidores trabajan en los niveles superiores del modelo OSI, y permiten que los usuarios no tengan que enfrentarse con la máquina Bastión, ya que su utilización camufla la existencia del Bastión, dando la sensación a los usuarios de que están trabajando directamente con la red externa. Este servidor actúa como cliente y como servidor, como servidor para los usuarios de la red interna, los cuales se deben comunicar con él, y como cliente por que se encarga de realizar las peticiones de servicio a las máquinas situadas en la red externa. De esta forma evitamos el problema de tener cuentas de usuarios en el cortafuego. La mayoría de servidores *proxy* cuentan con la posibilidad de registrar todos los sucesos que le afecten, lo que permitirá tener un mayor control sobre el origen y destino de las conexiones que realice o reciba. Entre los servidores PROXY más conocidos, podemos citar:

- Squid.
- FWTK.
- Sproxy.
- Wingate.

TOPOLOGÍAS

Hasta ahora hemos visto una serie de conceptos que nos van a permitir poder trabajar con las distintas configuraciones que puede adoptar nuestro esquema de seguridad. En este punto abordaremos cuál de las tres configuraciones se ajusta más a las necesidades de nuestro esquema de seguridad.

CONFIGURACIÓN 1

En la figura 3, podemos ver el esquema de una de las configuraciones básicas para defender nuestra red. Consiste en un cortafuego basado en el filtrado de paquetes, mediante el uso de un *router* dedicado. Mediante las reglas de filtrado hemos restringido el trafico tanto en un sentido como en otro. Este sistema es fácil de configurar y de mantener, pero presenta un punto débil debido a que basa nuestra política de defensa en la información que contienen los paquetes, dirección de origen, dirección de destino, puertos, protocolos, etc. Alguien podría cambiar parte de esta información, para intentar saltarse las reglas de filtrado y entrar en nuestra red. Supongamos la siguiente situación:

Nuestro sistema de seguridad coincide con el reflejado en la figura citada. Las siguientes, son consideraciones sobre los criterios de filtrado del tráfico que llega a la interfaz externa:

- Cualquier máquina de la red interna puede realizar una conexión TELNET a cualquier máquina de la red externa.
- Cualquier máquina de la red interna puede realizar una conexión FTP a cualquier máquina de la red externa.
- Se permite el acceso del dominio 150.100.* a un servidor HTTP con IP 195.0.0.10, perteneciente a la red interna.
- Se deniega cualquier paquete procedente del dominio 140.100.*, sea cual sea su destino y los puertos con los que trabaje.

TABLA 1. Tabla para la interfaz externa.

Regla		Acción	IP Orig.	IP Dest.	Puerto Orig.	Puerto Dest.
1	Permite	195.0.0.*	*	23	23	
2	Permite	195.0.0.*	*	*	21	
3	Permite	150.100.*	195.0.0.10	*	80	
4	Deniega	140.100.*	*	*	*	
5	Deniega	147.*	195.0.0.20	*	*	
6	Deniega	195.0.0.*	195.0.0.*	*	*	

- Se deniega cualquier conexión a la máquina 195.0.0.20, desde el dominio 147.

En la Tabla 1 para la interfaz externa tenemos expresadas todas las reglas anteriores. La regla 6 no pertenece a nuestro conjunto de reglas, más adelante veremos el motivo por el cual ha sido añadida.

A continuación se describen algunas consideraciones sobre los paquetes que llegan a la interfaz interna:

- Todos las máquinas de la red interna tienen acceso al servidor *HTTP* de la máquina 195.0.0.10.

- Se deniega cualquier intento de conexión utilizando el puerto 23 a cualquier máquina, sea cual sea su destino, tanto dentro como fuera de la red interna.

- Se deniega el acceso a todas las máquina que intenten conectar utilizando el puerto 80 de cualquiera de las máquina del dominio 130.130.*.

En la Tabla 2 para la interfaz interna, tenemos expresada el conjunto de reglas que se aplicarán a todos los paquetes que lleguen a la interfaz del *router* que da servicio a la red interna.

Ésta podría ser una de tantas políticas que seguir, pero esta política de filtrado de paquetes, no nos protege ante un ataque con **IP-Spoofing**. Este tipo de ataque consiste en alterar la información de los paquetes, para que dicho paquete pase las reglas de filtrado. Lo usual consiste en cambiar el campo IP de origen, por un IP perteneciente a la red interna. Vamos a suponer que alguien, en alguno de los numerosos *hosts* de la red externa cambia el campo que contiene la IP de origen del paquete, para que contenga una dirección perteneciente a la red interna.

El paquete llegaría a la interfaz externa del *router* y se le aplicarían las reglas, donde la primera dice que todo paquete cuya IP de origen pertenezca a la red interna tiene permitido el tráfico sea cual sea su origen. Por lo tanto, el *router* dejaría pasar el paquete y éste llegaría a su destino dentro de la red interna. Para evitar este tipo de ataques, añadimos una regla, la número 6, la cual denegará todo paquete, que llegue a la interfaz externa y cuyo IP de origen y destino pertenezcan a la red interna.

Si el *router* es atacado y no advertimos el ataque a tiempo, el atacante podría cambiar las reglas de filtrado y permitir que el tráfico fluyera en ambas direcciones a través del *router*.

CONFIGURACIÓN 2

La segunda configuración que comentado consiste en la implementación de una máquina Bastión. Esta máquina cuenta con dos dominios, es decir, posee dos interfaces de comunicación, y por lo tanto dos direcciones IP. Una dirección pertenece a la interfaz externa, es una dirección real, mientras que la dirección IP de la interfaz interna, no es una dirección real, sino ficticia, pudiendo tomar los valores que deseemos. para el ejemplo hemos elegido el dominio 111.0.0.*.

Para que esta configuración tenga sentido debemos asegurarnos, como hemos dicho anteriormente, que la opción de **IP Forwarding** se encuentre desactivada en el cortafuego. En caso contrario, encontraríamos que todos los paquetes que llegan por alguna de las dos interfaces son retransmitidos hacia el otro interfaz.

Como estamos controlando el tráfico en los niveles superiores del modelo OSI, para que tanto los usuarios de una red como de la otra se comuniquen con la otra red, primero deben entrar en el cortafuego y desde aquí realizar la conexión hacia su objetivo.

Este esquema evita los ataques del tipo **IP-Spoofing**, con direcciones de origen falsas, ya que los paquetes de la red externa nunca entrarán en la red interna. Pero este esquema presenta un problema, aún si cabe más peligroso y es que todos los usuarios que deseen acceder a una u otra red deben contar con una cuenta en el cortafuego. Teniendo en cuenta esto, llegamos a la conclusión que es más fácil crear un conjunto de reglas complejas que restrinjan el acceso a nuestra red interna mediante un *router* de filtrado, que concienciar a un grupo de usuario (muchos de los cuales no conocen los problemas de seguridad de los sistemas informáticos). Es decir, advertirles que deben utilizar palabras de paso para sus cuentas lo suficientemente buenas para que resulte difícil que sean adivinadas, así como no compartirlas con nadie.

Tabla 2. Reglas para la interfaz interna.

Regla	Acción	IP Orig.	IP Dest.	Puerto Orig.	Puerto Dest.
1 Permite	*	195.0.0.10	*	80	
2 Deniega	195.0.0.20	*	*	23	
3 Deniega	*	130.130.*	*	80	

CONFIGURACIÓN 3

El esquema de la figura 4 presenta una topología que recoge las mejores características de los dos esquemas anteriores: un *router* dedicado con filtrado y una máquina Bastión. El *router* se encarga de realizar una primera criba sobre los tres primeros niveles del modelo OSI, descartando todos aquellos paquetes que no cumplan con las reglas de filtrado. Los paquetes que cumplan las reglas de filtrado, se enfrentarán con la máquina Bastión. Esta máquina Bastión puede tener alguna de las dos configuraciones posibles, es decir puede ser o no, un servidor *proxy*.

Lo importante de este esquema es que se ha doblado el sistema de seguridad con respecto a las configuraciones anteriores, ya que si un intruso quisiera entrar de forma ilegal en nuestra red, primero tendría que atacar el *router* y después enfrentarse con el Bastión. Esta configuración no va a evitar el intento de entrada ilegal en nuestra red, pero vamos a tener un margen de tiempo más amplio que el que ofrecían las configuraciones anteriores, ya que si detectamos un ataque sobre el *router*, nuestra red aún está protegida con la máquina Bastión, por lo que tenemos un margen de tiempo, para actuar en consecuencia.

Estas son las tres configuraciones básicas de las que disponemos para construir un sistema de seguridad. Podemos combinar tantas configuraciones como deseemos, para, por ejemplo, construir un sistema con dos niveles, como refleja la figura 5. Uno exterior protegido por un *router* con filtrado y donde se ha situado un servidor *HTTP* cuya información pode ser considerada de poca importancia. El otro en un nivel interior con una máquina Bastión que se encargaría de proteger recursos que podríamos considerar más importantes, como podrían ser servidores de ficheros, etc.

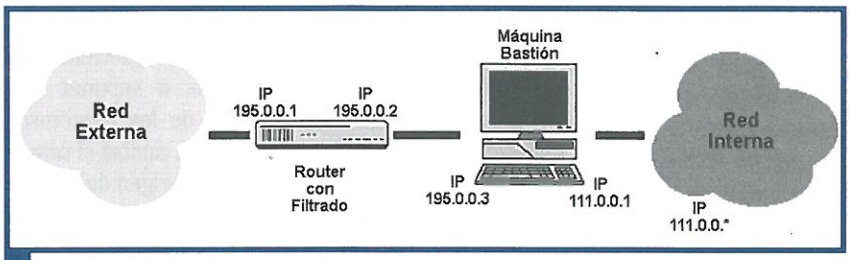


Figura 4 . Router y Máquina Bastión.

CONCLUSIÓN

En este artículo hemos dado a conocer las características más básicas para poder plantear los diseños de un sistema de seguridad robusto, pudiendo utilizar alguna de las topologías vistas aquí. No obstante hemos de destacar que en el mundo de la seguridad informática, la figura del administrador es primordial.

En el próximo artículo continuaremos la seguridad, veremos algunas de las herramientas software de las que podemos disponer a la hora de configurar un cortafuego, como son *TCP Wrapper*, el paquete *FWTK* (*FireWall Tool Kit*) de la casa *TIS*, y la herramienta SATAN que nos permite detectar posibles fallos de seguridad.

A continuación vamos a facilitaros una serie de direcciones que os podrán ampliar la información del tema tratado en el artículo:

Direcciones:
ftp://ftp.rediris.es/software/security/
SATAN
/net/satan/
TCPWrapper
/net/TCP-Wrapper
SSH
/net/ssh
Crack
/password/Crack
OPIE
/password/nrl-opie
S/Key
/password/SKey
DES
/crypt/des
IDEA
/crypt/idea

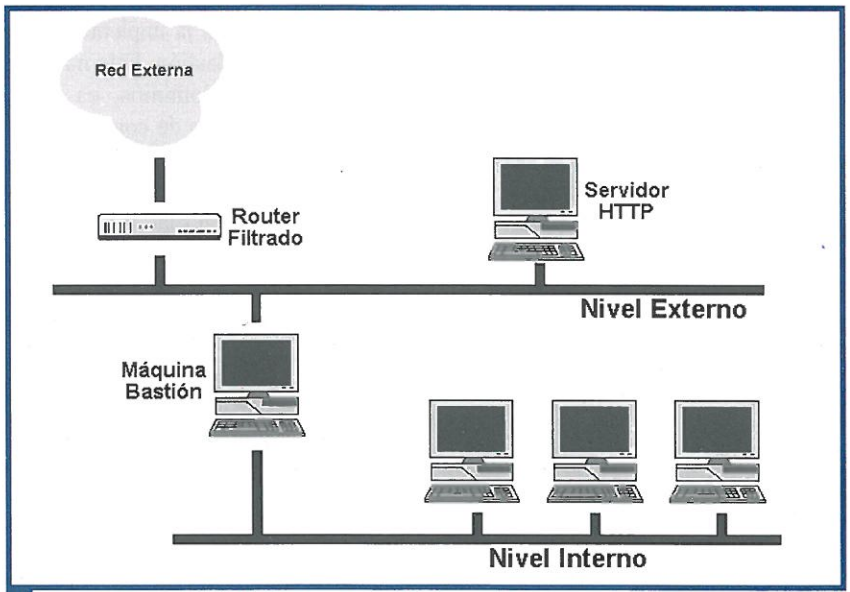


Figura 5. Esquema con dos niveles de seguridad.

Creación de componentes (y III)

Francisco Rosado Alacántara (fmra@redestb.es)

El mes pasado abordábamos la elaboración y diseño de un componente que simulaba un led. Deja[illegible]a, el icono que aparece en la paleta de [illegible]l resto.Lógicamente, es necesario mod[illegible]rece por defecto e incluir uno propio nues[illegible] queremos representar.

Para l
mejc
prese
quier
caso, por
land lo ofr
mienta C+
el Editor d
su ejecución
Image Edit
ols del menú

En prime[illegible] fichero de recur[illegible] para la aplicación, el cual permitirá entre otros muchos aspectos, personalizarla para que sea operativa en varios idiomas, cargando simplemente el fichero de recursos adecuado. En nuestro caso, el objetivo consiste en tener almacenadas las imágenes requeridas para nuestro componente. Dentro del editor de imágenes, seleccionaremos la opción File, después elegiremos New y a continuación Resource File. Para [illegible]*nap* utilizaremos las [illegible]ce, seguida de New y [illegible]ap

[illegible]piedades del bitmap, [illegible]ndo el botón derecho [illegible] la línea de la lista, [illegible]24 * 24 y VGA 16 [illegible]na ahora la opción [illegible]ú Resource intro- [illegible]LED para el obje- [illegible]este punto podremos utilizar las herramientas ofrecidas por el Editor de Imágenes para crear un dibujo, que posteriormente asociaremos al icono de nuestro componente. Por ejemplo, podríamos emplear un simple círculo verde.

Una vez que se ha diseñado el *bitmap*, lo deberemos guardar en el disco, teniendo en cuenta que tendremos que almacenarlo en el mismo lugar donde se encuentra el fichero *CPP* de nuestro componente De este modo tan simple hemos conseguido asociar un bitmap a nuestro componente, dándole sin duda una apariencia muy atractiva

CREACIÓN DE UN COMPONENTE SHAREWARE

Como programadores y usuarios de ordenadores, la mayoría de nosotros nos hemos encontrado con componentes y aplicaciones de tipo *shareware*, donde se presenta una pantalla inicial con aspectos relacionados con la versión, autor, duración, condiciones, El componente que vamos a diseñar se encarga precisamente de generar una pantalla como la comentada, es decir, un aviso que indica que la

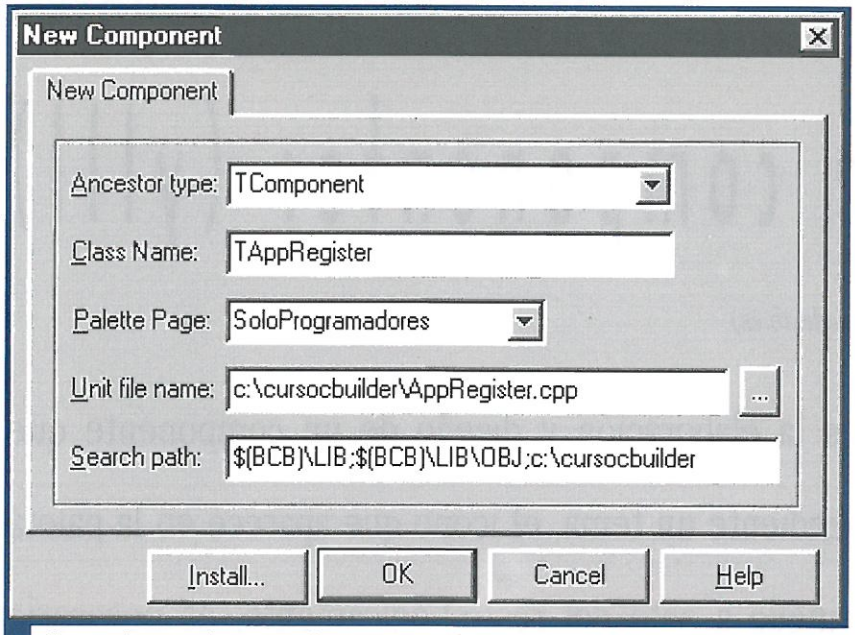


Figura 1. Asistente de creación de componentes

aplicación ejecutada es de tipo *shareware*. Por lo tanto, transcurrido un tiempo prudencial el usuario deberá registrarse para poder seguir usándola .

El objetivo del componente es:

- Realizar la 'legalización' del software en el momentos en que el usuario haya realizado el registro de la aplicación.
- Detectar al inicio de la aplicación si se encuentra resgistrad o nó, y si no lo está mostrar un aviso en relación con la situación.
- El registro de la aplicación será activado mediante la llamada a un método, definido en el componente. Dependiendo de si el registro de la aplicación ha sido satisfactorio o no, el método llamará a dos eventos que el programador debe desarrollar, personalizandolo para cada aplicación.

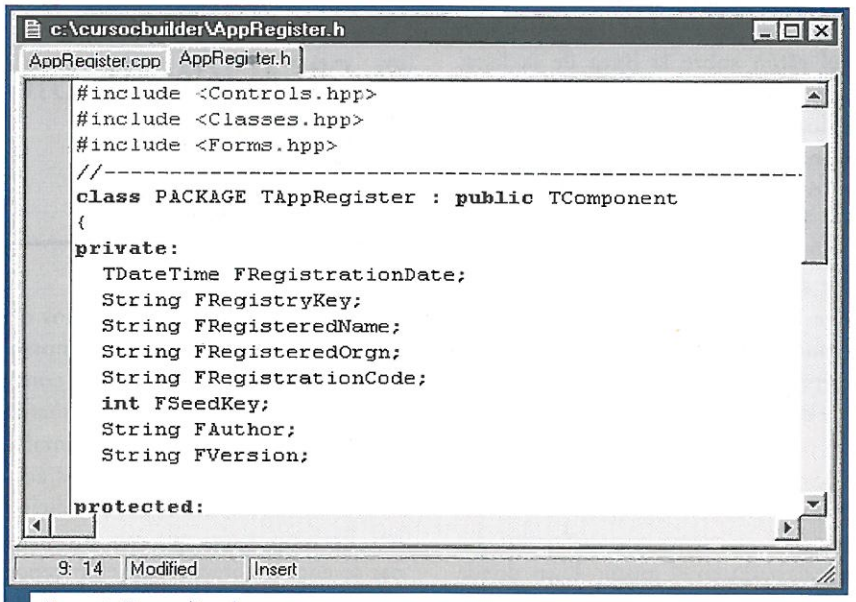


Figura 2. Nueva cláusula private.

CONSTRUCCIÓN DEL COMPONENTE

Para comenzar el desarrollo del nuevo componente arrancaremos C++ Builder, y cerraremos todas las aplicaciones y unidades que se encuentren abiertas. A continuación crearemos un nuevo componente utilizando el asistente de creación de la herramienta, en el que introduciremos como nombre el valor **TAppRegister**, siendo el componente **TComponent** del cual se heredará. En la figura 1 se pueden apreciar los valores necesarios para crear el esqueleto del nuevo componente.

Después de introducir los datos requeridos por el asistente, pincharemos sobre el botón **Ok**, de tal forma que obtendremos una base para desarrollar el componente comentado, tal y como vimos en anteriores entregas del curso.

Para continuar con el proceso, tendremos que abrir el fichero de cabecera asociado. Es importante recordar que para hacerlo visible tendremos que pulsar el botón derecho del ratón sobre la unidad y seleccionaremos la opción **Header File**. Añadiremos las propiedades que se van a visualizar en el Inspector de Objetos dentro de la cláusula *__published* y una definición de variables en la cláusula *private* tal y como recogen las figuras 2 y3.

Hemos creado una serie de propiedades de tipo *Read only* y otras no, que comentaremos con mayor detalle en las líneas siguientes

RegistrationDate, *RegisteresName*, *RegisteredOrgn* y *RegistrationCode* son propiedades de sólo lectura debido a que únicamente serán actualizadas cuando en realidad se produzca el registro de la aplicación.

Author y *Version* son datos relacionados con el autor del componente, así como la versión del mismo. La idea consiste en mostrar el nombre del mismo y la versión del componente que está siendo utilizado en el desarrollo de una aplicación.

La propiedad *RegistryKey* permite almacenar los datos del registro de la aplicación para poder verificarse en cada ejecución de la misma, empleándose el registro para la implementación.

SeedKey es una propiedad formada por un simple *integer* y cuya función consiste en controlar que un usuario no pueda registrar dos aplicaciones con un solo nombre. Es decir, si se usa el componente en dos aplicaciones, y se registra en una de ellas, no se pueda alegremente usar la otra, falsificando el registro.

Si ha seguido con atención el curso desde el inicio, seguramente te haya llamado la atención un aspecto. Las propiedades *Author* y *Version*, que van a reflejar el nombre del creador del componente y la versión respectivamente, han sido definidas como *ReadOnly*. Entonces, ¿cómo esposible que aparezcan en el Inspector de Objetos?

MAGIA, PROPIEDADES E INSPECTOR

Antes que nada, indicar que los valores de las propiedades citadas van a tomar un valor constante para el componente. Añadiremos las siguientes declaraciones en el fichero de cabecera:

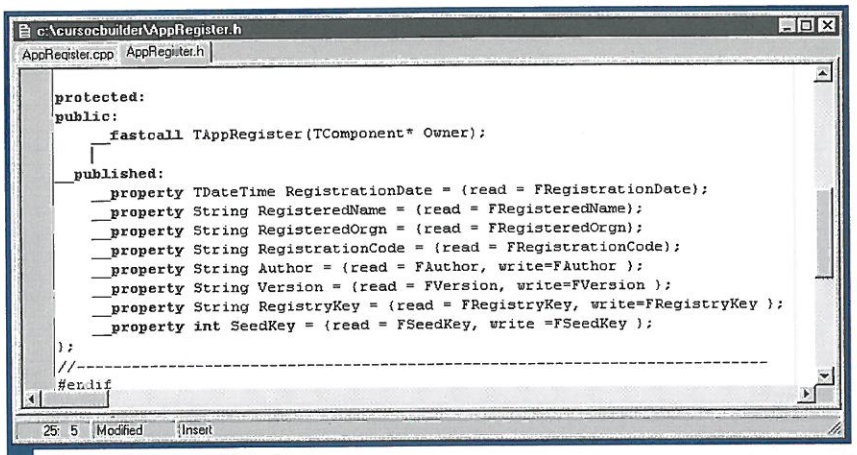


Figura 3 Cláusula _published.

```
Const string DefaultRegistryKey = "\\Softwa-
    re\\PredWare\";
Const string DefaultAuthor = "Paco Rosado";
Const string DefaultVersion = "V1.00";
```

.

Además tendremos que incluir la declaración de *RegistryKey* que utilizaremos más adelante, de tal forma modificaremos la definición de la clase para adaptarlas a nuestras necesidades. Introduciremos las líneas que aparecen en la figura 4.

Como consecuencia de las modificaciones efectuadas en el apartado anterior, también deberemos realizar las modificaciones pertienentes en la parte *published* del componente. Dichos cambios aparecen indicadas en la figura 5.

Dentro del fichero CPP, añadiremos las siguiente funciones:

```
void _fastcall TAppRegister::SetVersion(string
    pVersion)
{
 FVersion = DefaultVersion;
}
```

La función asociada al autor se realiza de forma análoga, de tal forma

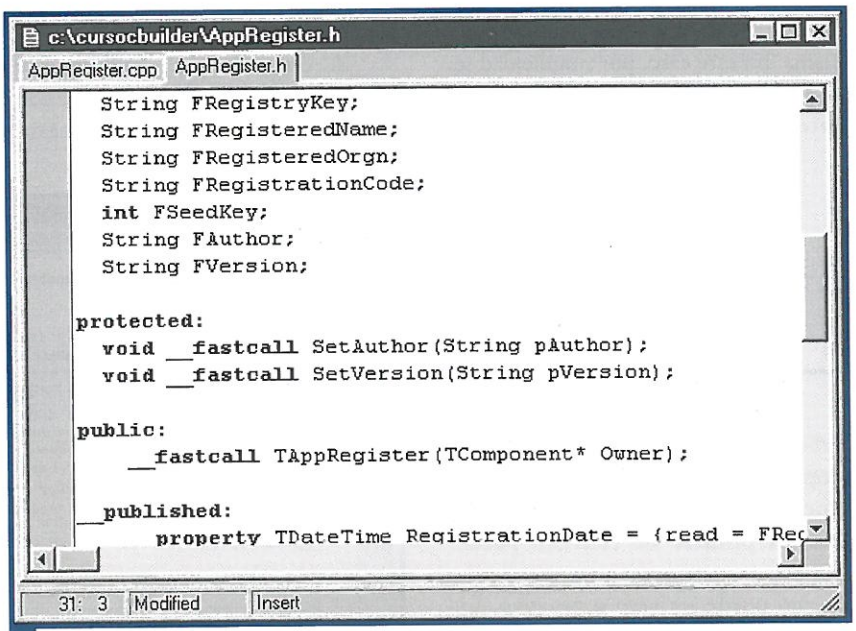


Figura 4. Parte protected del nuevo componente.

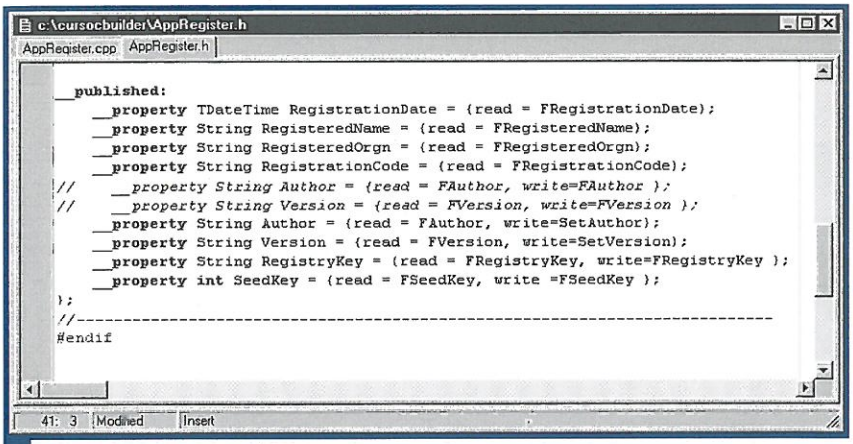


Figura 5. Nueva parte published

que al colocar las propiedas en la parte *publiched* será posible modificarlas desde el Inspector de Objetos. Así, cada vez que se cambie el valor de alguna de estas propiedades, se llamará a la función correspondiente (*SetVersion* en el caso anterior) y mediante la misma se asignará un determinado valor. En este caso concreto, el valor empleado será el que fue definido como constante en la cabecera de la aplicación. Dentro de la función será posible efectuar cualquier tipo de operación o validación, es decir, si se tratase de una propiedad en la que se almacena una fecha, podríamos validarla en la función comentada para no permitir que se introdujera un valor incorrecta para la misma. En este caso, por simplicidad se ha optado por asignar un valor constante definido al inicio del proyecto.

A VUELTAS CON EL COMPONENTE

Debemos añadir validaciones a otras propiedades del componente y más concretamente: RegistryKey y SeedKey. La primera de ellas no puede encontrarse vacío, mientras que SeedKey no puede ser cero o negativo. La forma de implementar estas opciones sería la misma que la empleada en las propiedades anteriores, de tal forma que tendríamos que implementar los métodos encargados de cambiar los valores cuando éstos no sean correctos. Añadiremos las siguientes definiciones de métodos dentro de la parte *protected*, junto con las definidas anteriormente.

```
void _fastcall SetRegistryKey(string Pregistry-
      Key);
void _fastcall SetSeedKey(int pSeedKey);
```

De igual forma tendremos que cambiar las propiedades que teníamos definidas, tal y como se muestra en la figura 6.

A continuación añadiremos el siguiente código en la ventana que contiene el código asociado a nuestro componente:

```
void _fastcall TAppRegister::SetSeedKey(int
      pSeedKey)
{
 FSeedKey 0 pSeedKey;
if (FSeedKey == 0) {
 FSeedKey = 1;
}
if (FSeedKey < 0) {
 FSeedKey = -FSeedKey;
 }
}

void _fastcall TAppRegister::SetRegistryKey
      (string pRegistryKey)
{
 FRegistryKey = pRegistryKey;
if (FRegistryKey == "")
  FRegistryKey = pRegistryKey;
if (FRegistryKey == "") {
 FRegistryKey = DefaultRegistryKey;
  }
}
```

Para asegurarnos de que los valores se cargan inicialmente de forma correcta, tendremos que efectuar las modificaciones pertinentes en el constructor, de la forma siguiente:

```
_fastcall TAppRegister::TAppRegister(TCompo-
      nent* Owner)
   : TComponent (Owner)
{
 FRegistryKey = DefaultRegistryKey;
 FRegisteredName = "Unregistered";
 FRegisterOrgn = "Unregistered";
 FRegistrationCode = "";
 FSeedKey = 1;
```

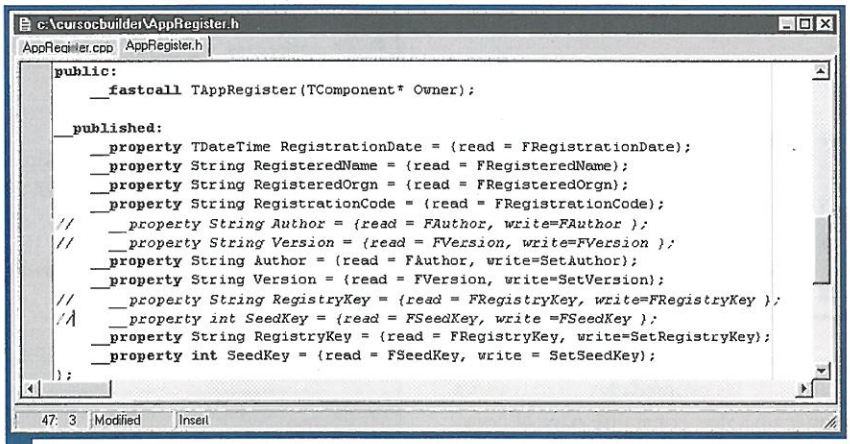


Figura 6. Nuevos propiedades del componente.

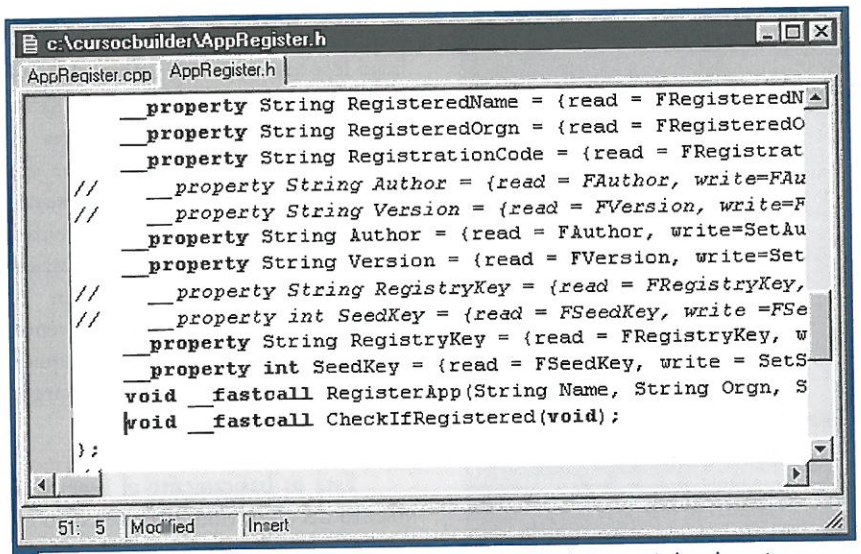


Figura 7. Definición de los métodos encargados de controlar el registro de la aplicación.

llama a *DeriveRegCode* para convertir el nombre de registro ofrecido en su correspondiente código de registro, de tal forma que si el valor obtenido concuerda con el código suministrado, el proceso de registro se considerará correcto. En este momento, los valores de las propiedades relevantes pasan a tener los valores de los parámetros de registro pasados. A continuación, los datos correctos son almacenados en el registro del sistema (Win95), dentro del lugar especificado en la propiedad *RegistryKey*.

En caso de que el proceso de registro sea erróneo, nos aseguraremos de que las propiedades más relevantes conservan unos valores adecuados.

```
FAuthor = DefaultAuthor;
FVersion = DefaultVersion;
}
```

EL TRABAJO DEL COMPONENTE

El objetivo del componente consiste en implementar dos métodos, de tal forma que uno chequee el estado actual de la aplicación (si ya ha sido registrada o nó) y otro que permita al usuario seleccionar información complementaria. En primer lugar, añadiremos las cabeceras necesarias para la inclusión de estos dos métodos, dentro de la parte *published*, tal y como se aprecia en la figura 7.

A continuación introduciremos el código relativo a los dos métodos citados en el listado 1.

Antes de seguir, resulta conveniente realizar algunas aclaraciones sobre el código del listado 1. El método *RegisterApp* recibe tres parámetros, mediante los que obtendremos los detalles sobre el registro necesarios para comprobar si todo es correcto. Se han definido algunas variables y se

Listado 1. Código del componente encargado del registro.

```
void __fastcall TAppRegister::RegisterApp(String Name, String Orgn, String Code)
{
TRegistry *MyReg; //Define puntero
String ActualCode;
TDateTime DateNow;
ActualCode = DeriveRegCode(Name);
if (ActualCode == Code) {
FRegisteredName = Name;
FRegisteredOrgn = Orgn;
FRegistrationCode = Code;
FRegistrationDate = DateNow.CurrentDate();
MyReg = new TRegistry; //Asigna el puntero
MyReg->RootKey = HKEY_CURRENT_USER; //Set Root key inside registry
if (MyReg->OpenKey(FRegistryKey, TRUE)) {
try {
MyReg->WriteString("Registered Name", FRegisteredName);
MyReg->WriteString("Registered Organisation", FRegisteredOrgn);
MyReg->WriteString("Registration Code", FRegistrationCode);
MyReg->WriteDate("Registration Date", FRegistrationDate);
}
catch (...) {
; //nada
}
}
MyReg->Free();
if (OnRegSuccess) {
OnRegSuccess(this, FRegisteredName, FRegisteredOrgn,
FRegistrationCode, FRegistrationDate, amManual);
}
}
else{
```

Listado 1. (Cntinuación)

```
FRegisteredName = "Unregistered";
FRegisteredOrgn = "Unregistered";
FRegistrationCode = "";
if (OnRegFailure) {
OnRegFailure(this, amManual);
}
}
}

void __fastcall TAppRegister::CheckIfRegistered(void)
{
TRegistry *MyReg; //Define puntero
String RegName;
String RegOrgn;
String RegCode;
String ActualCode;
TDateTime RegDate;
bool Registered = FALSE;
MyReg = new TRegistry; //Guarda eapcio y el puntero
MyReg->RootKey = HKEY_CURRENT_USER;
if (MyReg->OpenKey(FRegistryKey, FALSE)) {
try {
RegName = MyReg->ReadString("Registered Name");
RegOrgn = MyReg->ReadString("Registered Organisation");
RegCode = MyReg->ReadString("Registration Code");
RegDate = MyReg->ReadDate("Registration Date");
ActualCode = DeriveRegCode(RegName);
if (ActualCode == RegCode) {
FRegisteredName = RegName;
FRegisteredOrgn = RegOrgn;
FRegistrationCode = RegCode;
FRegistrationDate = RegDate;
Registered = TRUE;
}
catch (...) {
; //nada
}
}
MyReg->Free();
if (Registered) {
if (OnRegSuccess)  {
 OnRegSuccess(this, FRegisteredName, FRegisteredOrgn,
 FRegistrationCode, FRegistrationDate, amAuto);
 }
}
else {
if (OnRegFailure) {
 OnRegFailure(this, amAuto);
 }
}
```

El método check tiene como finalidad leer los datos del registro del sistema. Si no es capaz de leerlos, se llama al evento encargado de notificar el fallo, si está definido. Si la lectura del registro del sistema es correcta, sustituiremos los valores por los leídos desde el registro. Si además, coincide con el código almacenado en el *Registry*, entonces llamamos al evento *acierto* si está definido. En cualquier otro caso, llamamos al evento encargado de avisar del fallo.

Este es básicamente el funcionamiento del algoritmo, aunque como es lógico, se podría mejorar. En caso de efectuar alguna modificación, por favor, no dude en ponerla en conocimiento del autor de este artículo.

Durante el código anterior nos hemos saltado la definición del algunas funciones y métodos.

No existe una función llamada *DeriveRegCode* y no existe un evento definido para llamarla. *DeriveRegCode* es una función que, a partir de un nombre de registro, obtiene un código de registro, que es el que comparamos con el que tenemos en nuestro componente. Esta rutina debe ser añadida como un método de nuestro componente. Pero, ¿*protected* o *public*? Lo lógico es que la declaremos como *public* de modo que los desarrolladores de aplicaciones puedan escribir código que genere esos códigos de registro. Añadiremos la definición del método en la parte *published*, como se ve en la figura 8. El código asociado a ese método se muestra a continuación:

```
string _fastcall
      TAppRegister::DeriveRegCode(string
      Name) {
 int i;
 int RegCode = 0;
for (i=1; i<=Name.Length(); i++) {
   RegCode += Name[i] * FSeedKey;
 }
return (AnsiString(RegCode));
}
```

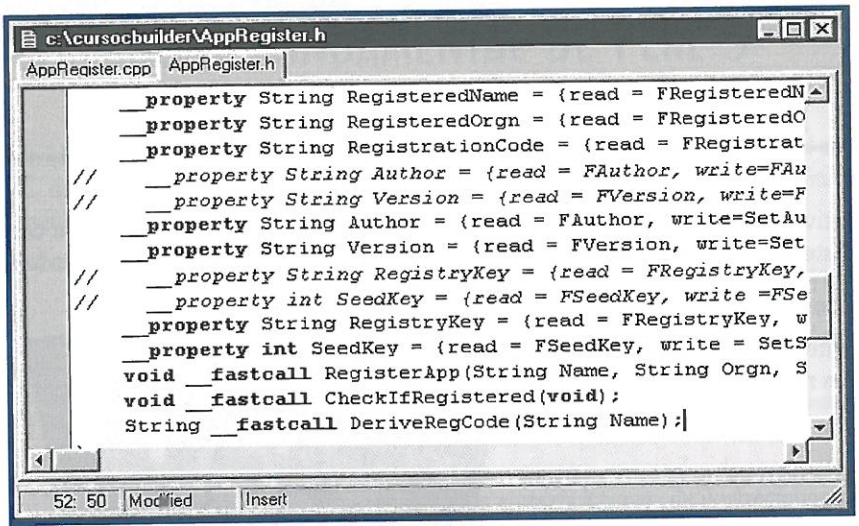


Figura 8. Nuevo método

El algoritmo usado para la validación del registro es el siguiente: multiplicamos el código ASCII de cada carácter pasado como parámetro, lo multiplicamos por el valor almacenado en *SeedKey*. Sumamos los valores obtenidos con todos los valores del nombre de registro pasado como parámetro y lo convertimos en una cadena. Pero lógicamente, es posible utilizar el algoritmo de encriptación deseado por el creador del componente.

AÑADIENDO LOS EVENTOS

En este punto tenemos que incluir dos eventos: uno para el éxito del registro y otro para el fallo. Tendremos que introducir el siguiente código, definiendo los tipos de eventos al principio del fichero de cabecera, antes de la definición de clases.

El *enum* inicial es un tipo enumerado para los dos posibles valores del *flag* que se pasan a los parámetros para saber de donde vienen, si de un registro o de un fallo. Además tenemos dos tipos definidos llamados *tRegSuccessEvent* y *tRegFailureEvent*, asociados con los eventos citados anteriormente.

```
typedef enum TRegModes (amAuto, amManual)
      TRegMode;

typedef void _fastcall (_closure
  *TRegSuccessEvent) (TObject
, String RegisteredName
, String RegisteredOrgn
, String RegistrationCode
, TDateTime RegistrationDate
, TRegMode RegMode);

typedef void _fastcall (_closure
  *TRegFailureEvent) (TObject
, TRegMode RegMode);
```

Añadiremos las definiciones de las propiedades de los eventos en la parte privada de la clase, tal y como se muestra a continuación.

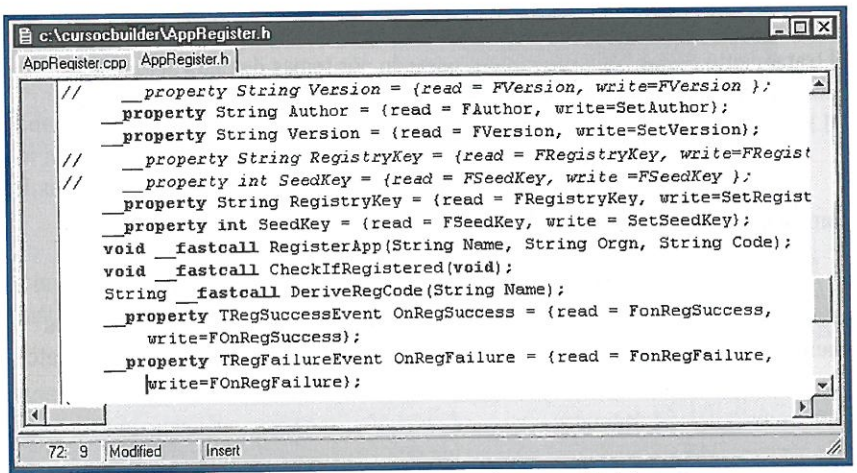


Figura 9. Definición de los eventos relacionados con el éxito o fallo del registro.

```
class PACKAGE TAppRegister:
public TComponent
 {
private:
  TDateTime FRegistrationDate;
  String FRegistryKey;
  String FRegisteredName;
  String FRegisteredOrgn;
  String  FRegistrationCode;
  int FSeedKey;
  String FAuthor;
  String FVersion;
  TRegSuccessEvent FOnRegSucces;
  TRegFailureEvent FOnRegFailure;
}
```

Y las definiciones en la parte *published* visibles desde el Inspector de Objetos quedarán tal y como se puede apreciar en la figura 9.

En este punto hemos terminado la creación del componente *shareware*, de tal forma que para un correcto uso del componente será necesario llamar al método *CheckIfRegistered* desde el código de nuestra aplicación.

Espero que el desarrollo de este nuevo componente haya sido de utilidad para los lectores, recalcando de nuevo que para cualquier duda o aclaración, no dudéis en poneros en contacto con el autor de este artículo en la dirección e-mail que encontrareis en la cabecera.

Correo del lector

En esta sección, los lectores de SÓLO PROGRAMADORES tienen la oportunidad de hallar respuesta a los problemas que puedan tener en cualquier tema relacionado con el desarrollo de las aplicaciones.

Pregunta

Hola, soy un asíduo lector de su revista. Hace poco que tengo una conexión a Internet y querría usarla desde mi sistema Linux (tengo instalada la distribución Eurielec Linux 98 que adjuntaba su revista). Seguí las instrucciones para conectarse del número 40 de su revista y, aunque desde consola establece la conexión, no consigo que Netscape reconozca la conexión. Lo he intentado con el kppp, pero me dice que la conexión acaba inesperadamente. El archivo de opciones y el script para la conexión son los que publicasteis (excepto la opción +ua, que no reconoce).

He intentado cargar pppd desde una terminal e iniciar mas tarde Netscape pero me da el siguiente error: "Netscape is unable to locate the server. The server does not have DNS entry". Me gustaría saber qué debo hacer para acceder a Internet desde Netscape en KDE.

Respuesta

Los mensajes de error que Netscape muestra en pantalla al establecer una conexión con el ISP (Servidor de Acceso a Internet) se deben a una mala configuración o creación de los scripts. Para resolver el problema, el primer paso a seguir es eliminar todos los scripts de conexión creados anteriormente. Deberían ser tres los eliminados, el script /etc/ppp/**options**, donde es guardada la información sobre la conexión, el /etc/ppp/**password** donde es guardado el login y la contraseña, y finalmente el /etc/ppp/**dial-isp**, donde es guardado el número del ISP. Los nombres de los scripts pueden variar así como el directorio donde se encuentran.

El segundo paso es crear el script /etc/ppp/**options** utilizando cualquier editor de textos, como podría ser vi. Seguidamente se debe copiar el siguiente texto tal y como está, sin eliminar la opción +ua, ya que debería ser reconocida. Una vez creado y guardado el script, se le debe aplicar la opción chmod 640 /etc/ppp/options.

```
#!/bin/sh
connect /etc/ppp/infovia
crtscts
modem
passive
+ua /etc/ppp/infoviappp
noipdefault
debug
defaultroute
asyncmap a0000
/dev/cuaX (Aquí se debe introducir el directorio
    donde el módem se encuentra)
38400 (Aquí se debe introducir la velocidad del
    módem)
```

El próximo paso es similar al anterior, también se deberá crear el script /etc/ppp/**infovia** y copiar el siguiente texto tal y como está. Es muy importante copiarlo exactamente igual, ya que un espacio no insertado podría hacer que el script no funcionara adecuadamente. Si se dispone de acceso directo al ISP, donde pone 055, deberá introducir el número de teléfono del servidor de acceso a Internet. Una vez guardado se le debe aplicar la opción chmod 744 /etc/ppp/infovia.

```
#!/bin/sh
/usr/sbin/chat -v "" atdt055 CONNECT ""
```

El tercer script a crear es /etc/ppp/**infoviappp**, deberá introducir su login en donde pone su_login y su contraseña en: su_contraseña. Al igual que el primer script, deberá aplicar la opción chmod 640 /etc/ppp/ infoviappp.

```
su_login
su_contraseña
```

Como último paso, deberá editar los scripts /etc/**rc.config** y /etc/**resolv.**

conf para asegurarse que la conexión se realiza con éxito. En el script rc.config, deberá buscar unas líneas parecidas a las siguientes y asegurarse de que ninguno de los IPADDR_ contiene una dirección IP, sólo deben estar las dos comillas (""). :

```
#
#IP Adresses
#
IPADRR_0=""
IPADRR_1=""
IPADRR_2=""
IPADRR_3=""
```

El siguiente paso es buscar la línea en donde se encuentra el siguiente texto:

```
SEARCHLIST="suservidor"
```

En esa línea deberá introducir el nombre de su servidor y su terminación. Por ejemplo, si su servidor es towercom.es, así debería quedar en rc.config:

```
SEARCHLIST="towercom.es"
```

La última línea a buscar es la siguiente:

```
NAMESERVER=direcciónip
```

En lugar de direcciónip, deberá introducir la dirección IP que su servidor proporciona.

Como último paso para completar el proceso tiene lugar, en el script /etc/-**resolv.conf**, deberá editar la línea donde pone search o domain e introducir el nombre de su servidor y su terminación, como podría ser:

```
search towercom.es
```

También deberá introducir la dirección IP asignada por su servidor en la línea nameserver. Un ejemplo sería:

```
nameserver 194.179.1.100
```

Si se ha seguido la creación, configuración y edición de todos los scripts, se debería poder establecer una conexión a un ISP con éxito. Para poder comprobar si todo funciona correctamente, se debe ejecutar pppd y luego utilizar uno de los servicios como puede ser WWW (con Netscape), FTP, E-mail, etc. Para finalizar la conexión se deberá ejecutar ps, buscar el proceso pppd y su número, y teclear kill -9 número del proceso pppd.

Pregunta

Hola amigos, soy un entusiasta de vuestra publicación y además de felicitarles quisiera que me solucionaran una gran duda para mí. Se trata de lo siguiente: he seguido vuestros artículos (y estudiado algún que otro libro al respecto) acerca de la programación en HTML y más precisamente la gestión de CGI (Common Gateway Interface).

El problema es que no sé cómo hacerlos funcionar. Los browsers (Microsoft Internet Explorer 3 y 4.0) o Netscape Navigator, interpretan los formularios HTML, pero no "disparan" los ejecutables referenciados por el botón de submit, en mi propia PC.

Quiero decir, como sólo dispongo de una PC con Windows 95 para desarrollar (y por supuesto, poquísimo dinero para comprar herramientas de software ni armarme una red), quisiera poder testear los mismos en una sola máquina, si ello es técnicamente posible. Cada vez que sale un HTTP Server (que vosotros publicáis en el CD-ROM), lo copio y obviamente, el server se queda "escuchando" en el port 80, pero los browsers parecen como que lo ignoran. Me gustaría alguna contestación, o respuesta personalizada a vuelta de correo (electrónico, claro), o también explicaciones generales o referencias a artículos ya publicados (que me parece que entiendo mal en algo). El "Personal Web Server" que viene con el FrontPage de Microsoft, cumple esa función, si es así, ¿hay alguna versión shareware o freeware equivalente que me podáis recomendar?

Respuesta

Estimado lector, para empezar debo insistir en que efectivamente necesitas de un software que haga las veces de servidor si quieres desarrollar y ejecutar programas CGI en tu propio PC. Cuando tengas ese servidor correctamente instalado y funcionando, podrás poner tus programas en un directorio que el servidor reserve para tal efecto, y estos se ejecutarán cuando desde el navegador del que los llames.

Lo programas los podrás hacer en C, Perl, Java o cualquier otro lenguaje dependiendo de las limitaciones que te imponga el servidor.

Bien, por otro lado parace que sí has intentado instalar un software servidor, aunque no con demasiado éxito. El Personal Web Server que viene con FrontPage es un ejemplo estupendo, y te puedo asegurar que funciona perfectamente. Intentaré explicarte brevemente qué debes mirar.

En primer lugar debes ir al Panel de control de Windows. Allí tendrá que aparecerte un icono llamado "Servidor Personal de Web". Si haces doble clic sobre él se abrirá una ventana donde podrás configurar el funcionamiento del servidor. Para empezar lo más importante en que debes fijarte es la dirección URL que el servidor va a intepretar como suya (p.ej. en mi caso es "http://default"). Así una vez que inicies el servidor, cuando teclees *http://default/miweb*, donde "miweb" es un sitio Web creado con Frontpage, irás a las páginas correspondientes. De esa forma "pasas" por el servidor, cosa que no harás si simplemente te limitas a cargar las páginas de un directorio cualquiera de tu PC.

FORMULA ONE JAVA 5.5

Applet Java para realizar hojas de cálculo en la web. Nos permite la publicación de hojas de cálculo personalizadas en la web.

Una vez puesta en la red es posible que el equipo cliente realice operaciones matemáticas en tiempo real con tan sólo modificar las celdas.
.

Su capacidad puede soportar más de 150 operaciones y dispone de más de 400 propiedades distintas para su configuración.

JAVA PERK 4.22

Entorno para la creación de animaciones Java. Gracias a esta aplicación es muy sencillo realizar elementos animados Java.

Es ideal sobre todo para crear presentaciones, textos animados, botones dinámicos, scrolls, efectos y cortinillas., así como para desarrollar la animación basta con seleccionar el elemento y aplicarle tantas propiedades como sean necesarias.

Precisa las runtime de Visual Basic 5.0.

PACMAN

Juego de comecocos en un applet Java. Poco hay que decir de uno de los juegos que más se ha resistido al paso del tiempo.

Esta vez se trata de un applet Java que puedes usar en tus páginas web para animar a las visitas. Dispone incluso de sonido. Muy divertido.

HTML

NAVROAD OFFLINE HTML VIEWER 2.20

Se trata de un navegador off-line con mínimos requerimientos de espacio. que está especialmente indicado para distribuir aplicaciones HTML a través de disquetes, ya que ocupa muy poco espacio. Puede ser muy útil para las emergencias.

SCRIBBLER 98 1.5

Éste es un editor de JavaScript, VBScript y HTML dinámico.

Se trata de uno de los primeros programas que se especializaron en el diseño y la creación de scripts de HTML dinámico en JavaScript y VBScript.

Dispone entre sus cualidades de grandes capacidades de edición y una librería de más de 30 scripts listos para utilizar.

DRUMBEAT 2.01

Excelente utilidad para la creación de sites con contenido dinámico de bases de datos.

Permite construir sitios que busquen y actualicen información en bases de datos ODBC para luego crear páginas de respuesta en tiempo real.

Se pueden crear efectos multimedia, incluir filtros DHTML, validación de formularios, etc.

OTROS

DELCHAR

Conversor de formularios Delphi de la versión 3.0 a la versión 1.0. El programa se encarga de eliminar propiedades del formulario como Font.CharSet, TitleFont.Charset, Flat y las Sets Property Scale para hacerlo compatible con la primera versión.

Su utilización es muy simple basta con seleccionar el directorio donde se encuentran los formularios Delphi y presionar sobre el botón "Convert" del programa.

FOREFRONT FOREHELP 3.0

Creación de ficheros de ayuda para sus desarrollos. Dispone de todas las herramientas necesarias para la creación de ficheros de ayuda para programas Windows.

El interfaz muestra el proyecto gráficamente con todas las asociaciones de hipertexto.

Dispone además de un procesador de textos propio, un editor de tablas y diccionarios. Puede utilizarse para crear ficheros .HLP tanto de 32 como de 16 bits.

ASM INTERPRETER 1.2

Ejecución de código ASM sin previa compilación. Muy sencillo de utilizar para cualquier usuario.

Dispone de editor/debugger que permite escribir y testear programas realizados en código ASM. Trabaja del mismo modo que un emulador de Visual Basic.

MULTILANGUAGE PACK 3.01 PARA DELPHI

En este caso se trata de un localizador multilingue de programas desarrollados en Delphi. Es una utilidad especializada en la traducción de programas desarrollados en Delphi.

Permite realizar la traducción tanto de los elementos del interfaz como de los propios componentes del programa. Para ello se sirve de la utilización de un diccionario especial que se encarga de desarrollar el trabajo.

NT SPECIFIC COMPONENTS PARA DELPHI 2.0/3.X/4.0 Y C++BUILDER 1.0

Componentes específicos Delphi y C++ Builder para Windows NT. Estos componentes ofrecen al desarrollador un camino nativo para la manipulación

de componentes de Windows NT desde sus programas.

ANOTHER PASCAL TO HTML CONVERTER

Conversión de ficheros Pascal a HTML. Con un interfaz muy simple y fácil de usar esta utilidad realiza una conversión rápida de ficheros Delphi en páginas HTML coloreadas. Basta con elegir el archivo Pascal de origen y el nombre del archivo .HTML de destino. El resto lo hace el programa.

HERRAMIENTAS/ UTILIDADES

COPERNIC 98 2.5

Herramienta muy potente para la búsqueda de información en Internet. El programa permite automatizar la búsqueda en más de 30 fuentes distintas simultáneamente. Puede hacerlo de la web, las news o de los directorios de E-mail. Utiliza AltaVista, Excite, Yahoo, HotBot, Infoseek, Lycos, Magellan, Open Text, WebCrawler y DejaNews entre otros.

MEDIDOR DE COSTOS TELEFÓNICOS 1.1

Una utilidad, en castellano, para controlar el gasto telefónico en conexiones a Internet.

Permite disponer de una factura personalizada con un listado de las llamadas realizadas, en el que se incluye la información del día, hora, tiempo consumido, etc.

SLOCK PROTECTION 1.0

Permite incluir restricciones a aplicaciones tipo shareware para obligar a que sean registradas. Muy útil para aquellos desarrolladores que pretendan desarrollar aplicaciones de tipo shareware o con algún tipo de restricción.

EXECUTOR 2.0W

Emulador de PC que permite ejecutar programas Macintosh. También permite acceder a unidades formateadas Mac y realizar operaciones con los ficheros que incluyen.

Ejecuta también las aplicaciones más populares de Macintosh como Microsoft Word, HyperCard, Claris Works, Quicken o PageMaker.

YEAR 2000 TEST

Programa que realiza el test de compatibilidad para el año 2000.

Analiza un sistema DOS o Windows 3.x/95/98 para ser capaz de definir si tiene alguna incompatibilidad ante el año 2000. Para ello lo que hace es reinicializar el PC en tres ocasiones y realiza una serie de tests que determinan si el reloj del sistema, la BIOS y el sistema operativo son totalmente compatibles.

REDES

LOCALES

BATCHMAIL32 `98

Sistema de mensajería interna para redes de área local. Trabaja bajo cualquier tipo de red local. Permite enviar y recibir correo, ficheros adjuntos e información sobre citas.

Posibilita la opción de mantener compartida la información diaria de la agenda.

FAXMAIL NETWORK 8.08

Programa servidor de envío y mantenimiento de documentos fax. Permite el envío o recepción de un fax a cualquiera de los componentes de una red local. Para el envío bastará con que realice la misma operación que cuando imprime un documento.

Posibilita la operación de mandar fax desde cualquier aplicación que permita imprimir.

WINPOP PLUS 1.2

Envío de mensajes en red con capacidad de control remoto. Permite la ejecución de aplicaciones remotas, el control de PCs a distancia y el envío de mensajes.

Es ideal para reemplazar el sistema de mensajes emergentes del propio Windows ya que dispone de características mucho más potentes, sin perder por ello la facilidad de uso y la operatividad.

DISTRIBUIDAS

GSM LAN GAME SERVER MONITOR 1.4

Monitor de servidores de juegos en red. Es un programa de 32 bits muy flexible que soporta cualquier tipo de red local. Soporta servidores de Quake, StarCraft, Hexen2 y Age of Empires.

No precisa de instalación, basta con descomprimir el fichero .ZIP y ejecutar la aplicación.

MDAEMON 2.7

Programa servidor de correo electrónico compatible SMTP y POP3. Permite la integración rápida del correo de electrónico Internet con el interno de una red local.

Dispone de firewall, editor de cuentas, posibilidades de control remoto, gestión de dominios y alias, soporte RAS, mailing lists, soporte RAW 3.0, programador de tareas, sistema antispam, etc.

NETWORK UNDELETE 1.1

Sistema de recuperación de archivos en Windows NT. Se trata de una aplicación muy interesante que permite la recuperación de ficheros corruptos o borrados bajo Windows NT 4.0.

Incluye en el mismo paquete la versión Administrator (servidor) y la Client (cliente).

@MAILGATE 2.48

Programa servidor de correo electrónico para red local. Un sistema de correo muy potente que actúa como puente entre el servidor POP3 del ISP y la red local. @MailGate realiza procesos preprogramados y permite aplicar filtros a los mensajes.

OTROS

MS DIAL-UP NETWORKING 40 BIT ENCRYPTATION UPDATE

Actualización del acceso telefónico a redes de MS. Otra nueva actualización del acceso telefónico, esta vez para implementar la encriptación de 40 bits y ofrecer así mayor seguridad.

DIAL-UP NETWORKING MONITOR 2.0

Gráficos y estadísticas sobre las conexiones remotas. Esta pequeña utilidad permite monitorizar y visualizar a través de gráficos todos los sucesos asociados a las conexiones a la Red.

Entre otros puede conocerse la velocidad de transferencia y los porcentajes de bytes enviados y recibidos.

DYNAMIP 3.31

Añade funciones Internet al sistema además de la gestión dinámica de direcciones IP, IPchat, WEBchat, HTTPscan, sincronización con el reloj del sistema, etc.

Es muy útil para usuarios avanzados de Internet que quieran mantener sus páginas con la dirección IP actualizada en todo momento.

MICROSOFT DIAL-UP NETWORKING 1.3

Actualización para Windows 95 de los componentes de acceso a redes. Incluye características adicionales al acceso telefónico a redes incluido en las primeras versiones de Windows 95.

Se encarga de añadir todas las mejoras para la gestión de dispositivos RDSI de ISDN 1.1.

Incluye soporte para adaptadores RDSI internos, soporte multilink para dos canales RDSI, programación del tiempo de conexión para automatizar las conexiones login no estándar y actualiza el soporte cliente PPTP.

MICROSOFT WINDOWS NT PPTP3 HOTFIX

Actualización que ofrece mejoras de seguridad a NT 4.0. Añade interesantes mejoras de seguridad para los servicios RAS/PPTP de Windows NT 4.0. Actualiza las mejoras incluidas en el Service Pack 3 de NT 4.0.

Incluye la nueva versión, más segura, de MSCHAP que ha sido implementada para las conexiones VPN.

MICROSOFT WINDOWS NT RRAS HOTFIX 3.0

Mejoras en el servicio de acceso remoto de Windows NT 4.0. Esta actualización requiere instalados los servicios de Routing y Acceso Remoto de NT Server 4.0.

Para instalar esta actualización es necesario descomprimir el fichero en un directorio temporal y posteriormente hacer doble clic sobre el fichero RRASFIXI.EXE.

DOCUMENTACIÓN /TUTORIALES

FILE FORMATS ENCYCLOPEDIA 2.0

Interesante enciclopedia sobre las características de los formatos de ficheros. Todo lo que debe saber un profesional de la programación sobre los formatos de los ficheros.

Entre los tipos de archivo desarrollados pueden encontrarse fichas sobre gráficos sonido, texto, animación, bases de datos, ejecutables, comunicaciones, etc.

SUPPORTSOURCE 2.02C WITH TECHCRAWLER

Programa especializado en mantener documentación técnica. Una herramienta muy interesante para desarrolladores. Permite gestionar la documentación técnica de nuestros propios desarrollos para poder realizar soporte técnico sobre el producto.

FUENTES DE LOS ARTÍCULOS

Programación de la SVGA (IV)
Estos son algunos listados de los programas incluidos.

Reconocimiento de voz (III)
Algunos ejemplos a los que se hace referencia en el artículo.

Fips 2.0
Utilidad para la creación y el mantenimiento de particiones Linux.

Universe Linux™

La alternativa que su empresa estaba esperando

Linux está en boca de todos: potencia, versatilidad, seguridad con mayúsculas, economía. Y todos se hacen las mismas preguntas. ¿Debo cambiar a **Linux**? ¿Es cierto que **Linux** es más fiable en la empresa que Windows NT, 98 o Windows 2000? ¿Es cierto que **Linux** puede convivir con Windows intercambiando información de forma transparente? ¿Puedo ahorrar con soluciones basadas en **Linux** hasta el 60% del precio en sus programas equivalentes de Microsoft? ¿Por qué más del 65% de los servidores de Internet corren bajo **Linux**? ¿Quién me dará el soporte y la garantía necesarios?

La respuesta a todas las preguntas se llama ***Universe Linux***

Universe Linux es la primera empresa de nuestro país dedicada exclusivamente a ofrecer soluciones basadas en **Linux** para usuarios y empresas: distribuciones **Linux**, utilidades, consultoría, instalaciones a medida, mantenimiento, formación... Todo lo que Vd. necesite y al mejor precio del mercado.

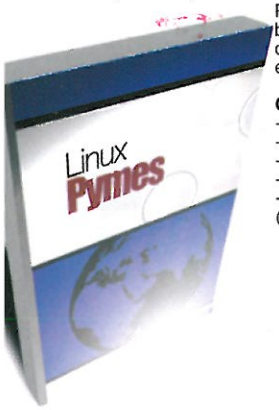
Puede que aún no sepa lo que ***Universe Linux*** puede hacer por Vd. Le invitamos a conocer nuestros productos con una demostración gratuita en nuestra sucursal de Madrid. Llámenos para concertar la demostración.
Infórmese sin compromiso llamando al **91-356 69 08** o visitando nuestra página web:

http://www.u-linux.com

Llame hoy mismo. *Por fin* en LINUX hay alguien que RESPONDE.

Linux PYMES

La distribución pensada para empresas
Pvp. recomendado 9.900 pts.



Por fin, la primera distribución creada para las PYMES: robusta, fiable, sencilla de instalar. Incluye: StarOffice, escritorio KDE, manual de instalación paso a paso y, como oferta de lanzamiento,GRATIS el programa LINUX FAX SERVER, valorado en 14.900 pts. (*)

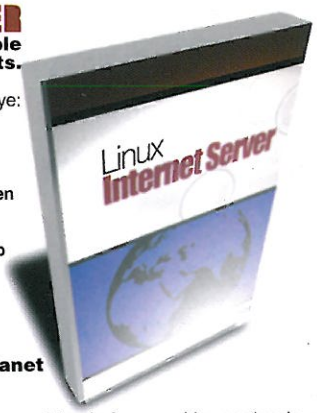
Características:
- Instalación en modo gráfico con pantallas de ayuda
- Configuración X completa antes de la instalación de paquetes
- Flexibilidad en los modos de instalación Server y Workstation
- Utilidades de automontaje
- XFree 3.3.5-3, Kernel 2.2.12-20, KDE 1.1.2, GTK, etc.

(*) Oferta válida hasta el 1 de abril del 2000.

FORMACIÓN
Cursos de Linux todos los niveles,
Básico
Administrativo
Seminarios

Linux INTERNET SERVER

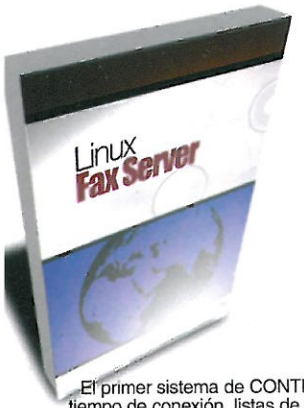
El servidor más fiable
Pvp. recomendado 49.950 pts.



El servidor de Internet pensado para su empresa incluye:
***Servidor de correo electrónico, páginas Web y FTP.**
***Servidor de proxy y caché.**
***Gestión y mantenimiento de listas de correo.**
***Alojamiento del Dominio y Dirección IP fija.**
***Gestión de usuarios y autentificación de los mismos en la entrada al sistema.**
***Control y visualización de usuarios conectados.**
***10 niveles de seguridad en el servidor de páginas Web local.**
***Programación de eventos de conexión con Internet.**
Ver demostración en http:// www.u-linux.com/demo

Linux FAX SERVER

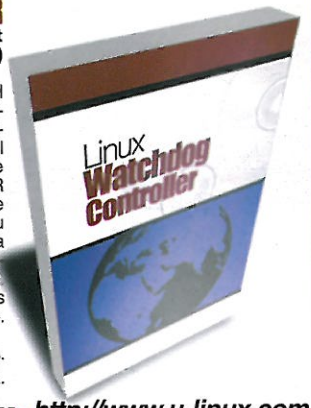
Ahorre enviando todos sus fax por la intranet
Pvp. recomendado 24.900 pts.



Se acabaron las colas para enviar un fax. El primer servidor de fax para Linux selecciona entre los tramos horarios más económicos de su operador de comunicaciones y se adapta a ellos, economizando en cada envío. Evita desplazamientos dentro de la propia empresa ahorrando tiempo y molestias al utilizar la intranet para acceder al servidor. Crea las estadísticas necesarias para analizar los gastos por cada usuario. Economía y comodidad garantizadas en sus comunicaciones.

LINUX WATCHDOG CONTROLLER

El guardián de su empresa en Internet
Pvp. recomendado 69.950 pts. (*)



El primer sistema de CONTROL TOTAL sobre Internet en su empresa. Limitación en el tiempo de conexión, listas de direcciones, chat, en general los puertos más representativos del sistema. Estadísticas de conexión por usuario, por sites accedidas, por distribución del tiempo según operador. Limitaciones individuales o por grupos de usuarios. El sistema corta la conexión cuando detecta que el usuario ha sobrepasado los límites de sus privilegios y le envía un mensaje al supervisor. LINUX WATCHDOG CONTROLLER es una herramienta imprescindible en la empresa moderna para el control del uso de Internet. El 90% de los empleados de las empresas con acceso a Internet desde su puesto de trabajo navegan con fines personales: chats, compras, etc. Esta herramienta erradica el problema realizando auditorías individualizadas de cada usuario, ya que almacena direcciones accedidas, tiempo de estancia, información recibida y enviada, etc. LINUX WATCHDOG CONTROLLER es una utilidad imprescindible para los Administradores del Sistema y Jefes de Personal. Pruébelo sin compromiso.

(*) Hasta 10 puestos de trabajo. Consultar según necesidades superiores.
Los precios no incluyen IVA.

Universe Linux *le brinda las mejores opciones para* **hacer negocio:**

Linux Training Center

Formación a todos los niveles

Si dispone de una academia de informática Vd. puede tener el LINUX TRAINING CENTER oficial de su población.
Universe Linux formará a su personal y le proporcionará todo el material didáctico necesario y la certificación LINUX TRAINING CENTER. Además, le incluiremos en nuestra Lista de Centros Autorizados en todas las acciones publicitarias promocionando su negocio. (*)
La más amplia gama de cursos para todos los niveles y necesidades en formación: Introducción a Linux, Administración de Sistemas, Instalación y configuración de Servidores web, Ofimática en Linux, etc.
Sistema de franquicias con número limitado de licencias por provincia.

(*) Más de 250.000 impactos cada mes en prensa especializada y general.

Linux Solution Provider

Una apuesta segura para emprendedores. Servicios en Linux

Si es Vd. socio o propietario de una empresa de Servicios Informáticos puede obtener para su provincia la representación en *exclusiva* de los productos **Universe Linux** y la certificación **LINUX SOLUTION PROVIDER** para su empresa de servicios, distribuyendo, instalando y dando soporte a la más amplia gama de productos Linux, con los mejores márgenes comerciales del mercado. Además, contará con el asesoramiento de nuestros técnicos y ayuda on line.
Un negocio de rentabilidad asegurada, sin cánones de entrada, compatible con su actividad y clientes habituales. No deje pasar la oportunidad e infórmese en el **91-356 69 08.**

Linux Developer

La mejor opción para desarrolladores

Universe Linux ofrece a los profesionales independientes del mundo de la programación la posibilidad de adquirir todos los conocimientos para trabajar en el mundo Linux. Incluye el material didáctico y el software necesario a un precio excepcional. Sólo para desarrolladores.

Y también: STAROFFICE, APPLIXWARE, ANTIVIRUS, bases de datos relacionales, servidores de ficheros, y mucho más.

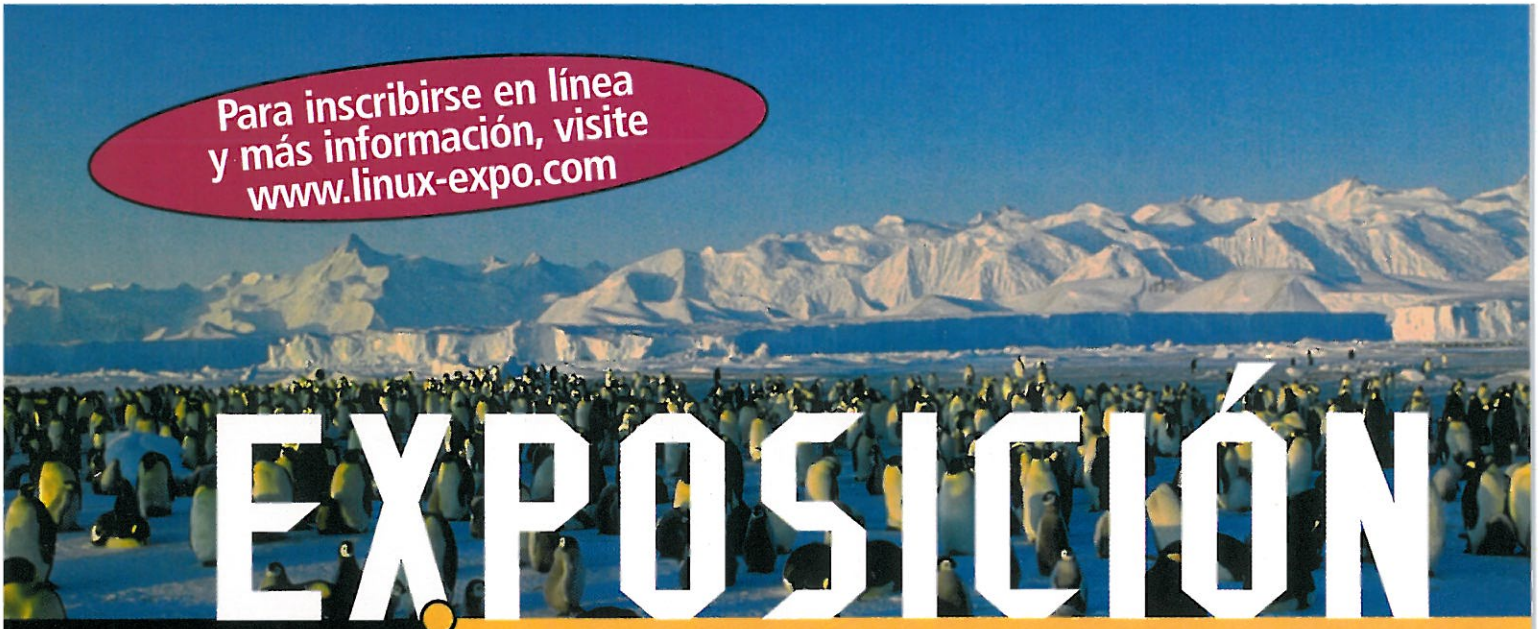
Distribuidores

Universe Linux S.L. España ofrece las mejores condiciones para distribuidores, con descuentos desde el 25% hasta el 40% según tramos de compra. Infórmese sin compromiso en el teléfono: (91) 356 69 08.
Si desea solicitar una demostración gratuita de alguno de los programas, póngase en contacto con nosotros.

UNIVERSE LINUX, LINUX SOLUTION PROVIDER, LINUX TRAINING CENTER, LINUX PYMES, LINUX INTERNET SERVER, LINUX WATCHDOG CONTROLER, son marcas registradas de U-LINUX, S.A. España.

WINDOWS NT, WINDOWS 98, WINDOWS 2000, MICROSOFT son marcas registradas de MICROSOFT CORPORATION.

rse Linux S.L. • C/ Conde de Vilches 21 • 28028 MADRID • Teléfono: (91) 356 69 08 • Fax: (91) 355 28 95 • http://www.u-linux.com



EXPOSICIÓN

CONFERENCIAS

El primer encuentro para soluciones bajo Linux y software libre.

26 y 27 de abril de 2000: de las 9.30 a las 19.30 horas.

LINUX EXPO

PALACIO DE CONGRESOS
MADRID
26-27 DE ABRIL 2000

NUESTROS SOCIOS.

SuSE LINUX
barrapunto.com
Hispa Linux
cyperus
AFUL
Prens@ Técnic
LINUX JOURNAL
LINUX ACTUAL
SuSE LINUX

OBJETIVOS 2000

3000 visitantes • 70 expositores
500 oyentes • 12 conferencias
1 sesión inaugural

CPP MARKETING SERVICES

Para contactarnos : SKY EVENTS : 12 av. de Corbéra • 75012 • PARIS • FRANCE
Tél. : (33) 1 43 45 80 80 • Fax : (33) 1 43 45 91 81 • E-mail : info@linux-expo.com

PEDIDO DE INFORMACION

Deseo recibir:

- ❑ Información para exponer en la Linux Expo Madrid
- ❑ invitación visitante.

Apellido:
Nombre:
Empresa:
Cargo:
Dirección:

Código postal: **Pais:**
Ciudad:
Teléfono:
Fax:
E-mail:

O SIMPLEMENTE ADJUNTE SU TARJETA DE VISITA.